Technical Report 352

Qualitative and Quantitative Knowledge in Classical Mechanics

Johan De Kleer

MIT Artificial Intelligence Laboratory

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AI-TR-352
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: Qualitative and Quantitative Knowledge in Classical Mechanics

5. TYPE OF REPORT & PERIOD COVERED: Technical Report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Johan de Kleer

8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0643

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS:
Advanced Research Projects Agency
1400 Wilson Blvd
Arlington, Virginia 22209

12. REPORT DATE: December 1975

13. NUMBER OF PAGES: 120

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:
Office of Naval Research
Information Systems
Arlington, Virginia 22217

15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Artificial Intelligence
Problem Solving
Qualitative Knowledge
Qualitative Mechanics
Quantitative Knowledge
Representation of Knowledge
Common Sense Knowledge
Planning
Physics
Symbolic Mathematics

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report investigates what knowledge is necessary to solve mechanics problems. A program NEWTON is described which understands and solves problems in a mechanics mini-world of objects moving on surfaces.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advance Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

QUALITATIVE AND QUANTITATIVE KNOWLEDGE

IN CLASSICAL MECHANICS

by

Johan de Kleer

Massachusetts Institute of Technology

December 1975

Revised version of a dissertation submitted to the Department of Electrical Engineering on August 15, 1975 in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

This thesis investigates what knowledge is necessary to solve mechanics problems. A program NEWTON is described which understands and solves problems in a mechanics mini-world of objects moving on surfaces.

Facts and equations such as those given in a mechanics text need to be represented. However, this is far from sufficient to solve problems. Human problem solvers rely on "common sense" and "qualitative" knowledge which the physics text tacitly assumes to be present. A mechanics problem solver must embody such knowledge. Quantitative knowledge given by equations and more qualitative common sense knowledge are the major research points exposited in this thesis.

The major issue in solving problems is planning. Planning involves tentatively outlining a possible path to the solution without actually solving the problem. Such a plan needs to be constructed and debugged in the process of solving the problem. Envisionment, or qualitative simulation of the event, plays a central role in this planning process.

Thesis Supervisor: Ira P. Goldstein
Title: Assistant Professor of Electrical Engineering and Computer Science

Acknowledgements

I would like to thank those people who encouraged this research. It was Seymour Papert who first suggested exploring qualitative mechanics to me. Without Ira Goldstein the thesis would never have been completed. Gerald Sussman, with his insight into mechanics and programming, was a great help. This work would not have been possible without extensive discussions with researchers at the MIT A.I. Laboratory, MIT Logo Laboratory, MIT Mathlab group and BBN. Especially, I would also like to thank: John Seely Brown, Hal Abelson and Andy di Sessa. I would like to thank Suzin Jabari for proof reading and drafting the diagrams.

TABLE OF CONTENTS:

1.0 INTRODUCTION

1.1 Introduction

This thesis is an investigation into problem solving in mechanics. A program NEWTON will be described which solves problems in the mechanics mini-world of "roller coasters" (the kinematics of objects moving on surfaces). We exclude from this investigation the issues of linguistically comprehending the natural language statement of the problem, analyzing the visual scene or manipulating the mathematical expressions. Instead we are interested in what reasoning takes place in what happens after the problem has been "understood" until a solvable set of mathematical equations has been produced. The mathematical expertise is based on routines culled from MACSYMA <Mathlab *et.al*, 74> and the visual and linguistic difficulties are evaded by a simple problem description language which describes the scene and presents the question.

One problem in the roller coaster mini-world is the well known loop-the-loop problem:

An object of mass m starts from rest and slides along a loop-the-loop of radius R as shown in the figure. What should the initial height z be so that the object successfully completes the loop-the-loop?

NEWTON would be presented this problem as:

```
(DEFINE-PROBLEM LOOP-THE-LOOP
       (SCENE (C1 CORNER * S1 -1)
              (S1 SEGMENT C0 C1 INCLINE PLUS 1)
              (C  CIRCLE 0 (C2 CORNER S1 ?S 0)
                           0 (X  CORNER ?S * -1)
                           POSITIVE NIL 1 4 R))
       (PLACE C1 ON ABOVE Z)
       (ASSIGN (HEIGHT S1) *)
       (QUERY (REACH X)))
```

The resulting solution NEWTON gives is:

globalconstraints:

$$(\text{HEIGHT-S1} - \frac{5\ R}{2}) > 0$$

One immediate place to look for information about the kind of knowledge involved in simple mechanics is the introductory text. At least at first glance these texts appear to outline the kinds of knowledge required to a sufficient degree to be formalized very easily. The difficulty appears when we observe how a student comes to acquire a competence at mechanics from studying this text. The student arrives with a vast amount of common sense knowledge about mechanics and problem solving and his competence comes from assimilating the new information into an already existing framework. Close examination of the physics text reveals an implicit assumption on the authors part that this framework be present. This pre-physics knowledge forms the basis and the vast bulk of the student's general physics knowledge. Newly acquired pieces of knowledge can only be understood against this background.

Classical mechanics has a well worked out mathematical formalism, however, people also have a great deal of intuition about mechanics and use that intuition to solve problems. Interaction between this quantitative or more mathematical knowledge and more qualitative intuitive reasoning plays a fundamental role in problem solving. NEWTON embodies both these kinds of knowledge. For example, qualitative knowledge is used to guide more formal mathematical manipulation and mathematical peculiarities which arise in the formal solving process are reinterpreted in qualitative terms.

In NEWTON this notion of qualitative vs. quantitative is further refined. Four basic aspects of mechanics problem solving are identified: question answering, envisioning, planning and quantitative reasoning. Question answering involves identifying the kind of question asked and generating a high level meta-strategy to deal with that question type. Envisioning involves intuitively visualizing the event in order to produce a qualitative description of the event. Planning takes the high level meta-strategy provided by the question answering and the qualitative

description provided by the envisioning to produce an explicit plan for the quantitative reasoning to follow. Quantitative reasoning employs the mathematical expertise to solve the relevant equations and these resulting equations are interpreted by the planning and question answering knowledge to decide how to proceed.

Most of the interaction is at the level of one kind of knowledge identifying what parts of the problem it cannot deal with and relegating this unsolved subproblem to another kind of knowledge. It can happen, and often does happen, that no new subproblems are discovered and the problem can be solved completely. For example, the question asking whether an object which is supported will fall can be answered without resorting to quantitative reasoning.

1.2 Relationship to Classical Science

One obvious place to look for both a description about the content of science and the representation of its knowledge is science itself. Such descriptions, unfortunately, lack good computational models. The development of such computational models is the one of the fundamental purposes of this thesis and is one of artificial intelligence's major contributions to science and education.

Most sciences are described in some mathematical formalism and any information which cannot be described in the formalism is usually poorly or never stated and left up to "intuition." This distinction between the mathematical knowledge and the intuition seems to be important and this research will articulate the distinction more clearly. We have termed this intuitive knowledge the qualitative, and knowledge involving recourse to mathematics and equations the quantitative. Not surprisingly then science suggests formalisms for the quantitative knowledge (i.e. mathematics), but nothing for the qualitative knowledge. This apparent dichotomy between quantitative and qualitative knowledge was the initial insight into this research.

We can still use this dichotomy to get valid insights into the structure of mechanics knowledge. This dichotomy will be examined in the acquisition of mechanics knowledge by students and the development of mechanics in history. In people there is a great resistance to using

quantitative knowledge. The solution is always attempted qualitatively and only if that breaks down is mathematics used. As is so often the case, the historical development of mechanics is very similar to its development in people. Mechanics was more or less understood qualitatively as far as recorded history; however, the quantitative understanding of mechanics did not really get developed or used extensively until the middle ages. Galileo and later Newton were the major figures in the completion of its development. Ernst Mach discusses this in *The Science of Mechanics* <Mach, 60>. The development of mechanics is described as passing through the three periods of observation, deduction and formalization. The last period in which all the quantitative knowledge of mechanics now lies is characterized as: "Here it is sought to put in a clear compendious form, or system the facts to be reproduced, so that each can be reached and pictured with the least intellectual effort." From a practical, educational, and historical viewpoint the quantitative knowledge comes last and is used as a last resort. It is, however, the natural outgrowth of the qualitative knowledge.

Unfortunately, that is all that science really has to say about mechanics knowledge. Even more unfortunate, what science call "formalization" is equivalent to being expressible in mathematical equations. This mathematical formulation is at great variance from what Mach claims for it: "... so that each [fact] can be pictured with the least intellectual effort." The search will have to look elsewhere than science to discover what is contained in qualitative knowledge.

1.3 Relationship to Previous Research

Much of current artificial intelligence research into the representation of knowledge is relevant to this thesis. The current interest in the "chunking of knowledge" influences this research <Minsky, 74>. Only a few researchers have, however, attempted to represent physical events and solve equations about them in the sense this thesis does.

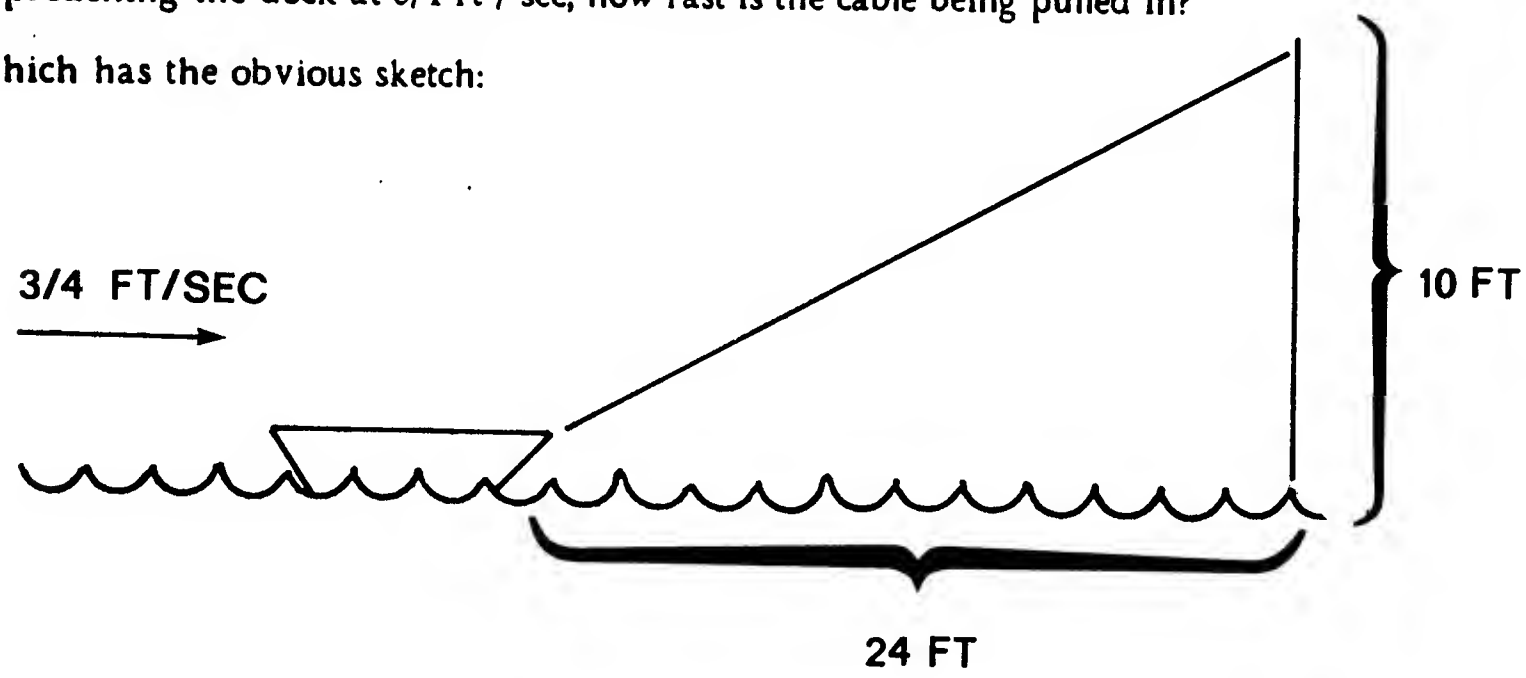
One such investigation was Bobrow's work with STUDENT <Bobrow, 68>. This was a program which could solve algebra word problems. Its Deductive Model solves the problems once they have been transformed from one's original English formulations. The major effort in this

work was in the transformation, not in the Deductive Model which was in essence very simple.

A later work by Charniak was CARPS <Charniak, 68> which could solve calculus rate problems expressed in English. Charniak used a more sophisticated description for events than Bobrow, but the major emphasis was still in the understanding of natural language. Charniak does, however, recognize the necessity for common sense knowledge. As an example he discusses a problem which on the surface appears solvable by CARPS, but which in fact is not:

"A barge whose deck is 10 ft below the level of a dock is being drawn in by means of a cable attached to the deck and passing through a ring on the dock. When the barge is 24 ft from and approaching the dock at 3/4 ft / sec, how fast is the cable being pulled in?"

Which has the obvious sketch:



CARPS assumes that "approaching the dock" means in direction of the pulling cable. It does not contain enough knowledge to be able to determine what "approaching the dock" means. CARPS neither knows that cables flex or that gravity constrains the barge to move along the surface of the water.

This problem points out the basic distinction between CARPS and NEWTON. CARPS uses general natural language strategies to set up equations using little or no real world knowledge. NEWTON, on the other hand, employs the strategy outlined in the previous section of first attempting to solve the problem using real world knowledge alone and only if that fails using that failure to guide further quantitative analysis. As a consequence of using real world knowledge extensively NEWTON can only solve problems from a very much smaller world than CARPS.

Within those areas that NEWTON knows something about it will be able to solve much harder problems and display far more common sense.

More recently, Charniak pursued an investigation into representing elementary electrostatics knowledge <Charniak, 71> (chapter one of <Purcell, 65>). Although the interest was again to develop "a firm semantic basis for a natural language understanding program", this work went further than the previous investigations in that he introduced a great deal more structure to the understanding. Many of the observations he made in his paper parallel ideas presented here, particularly the idea of a model. His arguments failed to be convincing since his domain and knowledge structure were so limited that there was rarely any choice in what to do next. He also failed to recognize the importance and necessity of qualitative knowledge.

2.0 A SURVEY OF MECHANICS KNOWLEDGE

2.1 Protocol Scenarios

This chapter is a survey of some of the basic kinds of knowledge used in solving mechanics problems. This will be done by a sequence of three worked out problems. Each problem is chosen to demonstrate a particular kind of knowledge. The purpose of these scenarios is to lay a framework for the fundamental ideas. With such a framework the later chapters which explain in much greater detail the kinds of knowledge represented and how NEWTON actually represents them can be read almost independently of each other. Chapters 3, 4, 5 and 6 present the relevant details. Chapter 7 summarizes the results.

The style of each scenario will be a statement of the mechanics problem which, except in the first case, is a standard college freshman physics problem, a possible protocol of how a student might solve the problem and then an analysis of that protocol to study the kinds of knowledge involved. Although some details of how NEWTON solves the problems are given, the later chapters will explain NEWTON's reasoning in far more detail.

The protocols given are hypothetical and no student would probably give such protocols. There is much more detail than the typical student might give; however, the basic structure seems correct. The difficulty is that an actual student will always direct his protocol at the level of expertise he expects from the protocol taker or himself and for the purposes of this discussion we are interested in all levels.

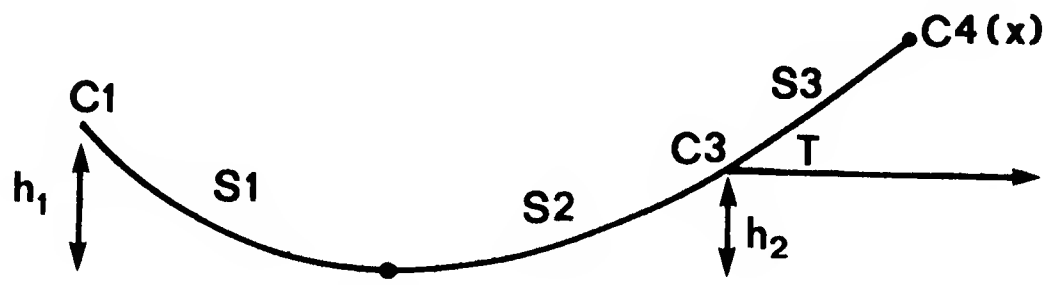
There is of course the question about what protocols, even hypothetical ones, are good for. It is clear that in people much more goes on in the problem solving process than is actually captured in the protocol and the protocol may in fact only be a sequence of after the fact rationalizations.

First, this research is not interested in modeling human behavior to the extent of duplicating the same difficulties the student gets into, that is, not interested in modeling bugs. Second, the protocols are reasonable in the sense they are able to communicate the ability to solve the problem. Last, the reasoning is logically correct and could be formally carried out. Whether or not the protocols are rationalizations they do express, at least at a general level, knowledge that must be

present to be able to generate the protocol (or rationalization) and furthermore, it would be a step forward in expert problem solving if NEWTON could solve these problems in the way these protocols outline.

The description for NEWTON outlined in the subsequent chapters will show how the following scenarios can be handled. Currently, however, the implementation of NEWTON in MACLISP at the MIT AI Lab can only handle the first two problems as the knowledge about inferencing and functional dependencies is not complete enough to handle comparison problems.

2.2 A Sliding Problem



A small block slides from rest along the indicated frictionless surface. Will the block reach the point marked *X*?

As is the case with most physics problems, this problem has many possible solution methods. The one presented here is not necessarily the best one. It has been chosen to illustrate the knowledge about envisioning and equations. Although NEWTON can solve the problem in the method described in the protocol, it would choose another "better" solution. These other solutions for this problem will be discussed at the end of this section.

"The block will start to slide down the curved surface without falling off or changing direction. After reaching the bottom it starts going up. It still will not fall off, but it may start sliding back. If the block ever reaches the straight section it still will not fall off there, but it may change the direction of its movement. To determine exactly whether the block reaches *X* we must

study the velocity of the block as it moves along the surface. The velocity at the bottom can be computed by using conservation of energy:

$$v_1 = (2\ g\ h_1)^{1/2}$$

Similarly using this velocity and conservation of energy we can set up an equation which can be solved for the velocity (v_2) at the start of the straight section:

$$1/2\ m\ v_2^2 = 1/2\ m\ v_1^2 - m\ g\ h_2$$

If the solution for v_2 is imaginary, we know that the straight segment is never reached. At the straight section we could use kinematics to find out whether the block ever reaches X. The acceleration of the block along the surface must be:

$$a = g\ sin\ T$$

The length of the straight segment is $L / cos\ T$, so using the well known kinematic equation relating acceleration, distance and velocities:

$$v_3^2 = v_2^2 - 2\ L\ g\ tan\ T$$

Again if v_3 is imaginary, X is not reachable."

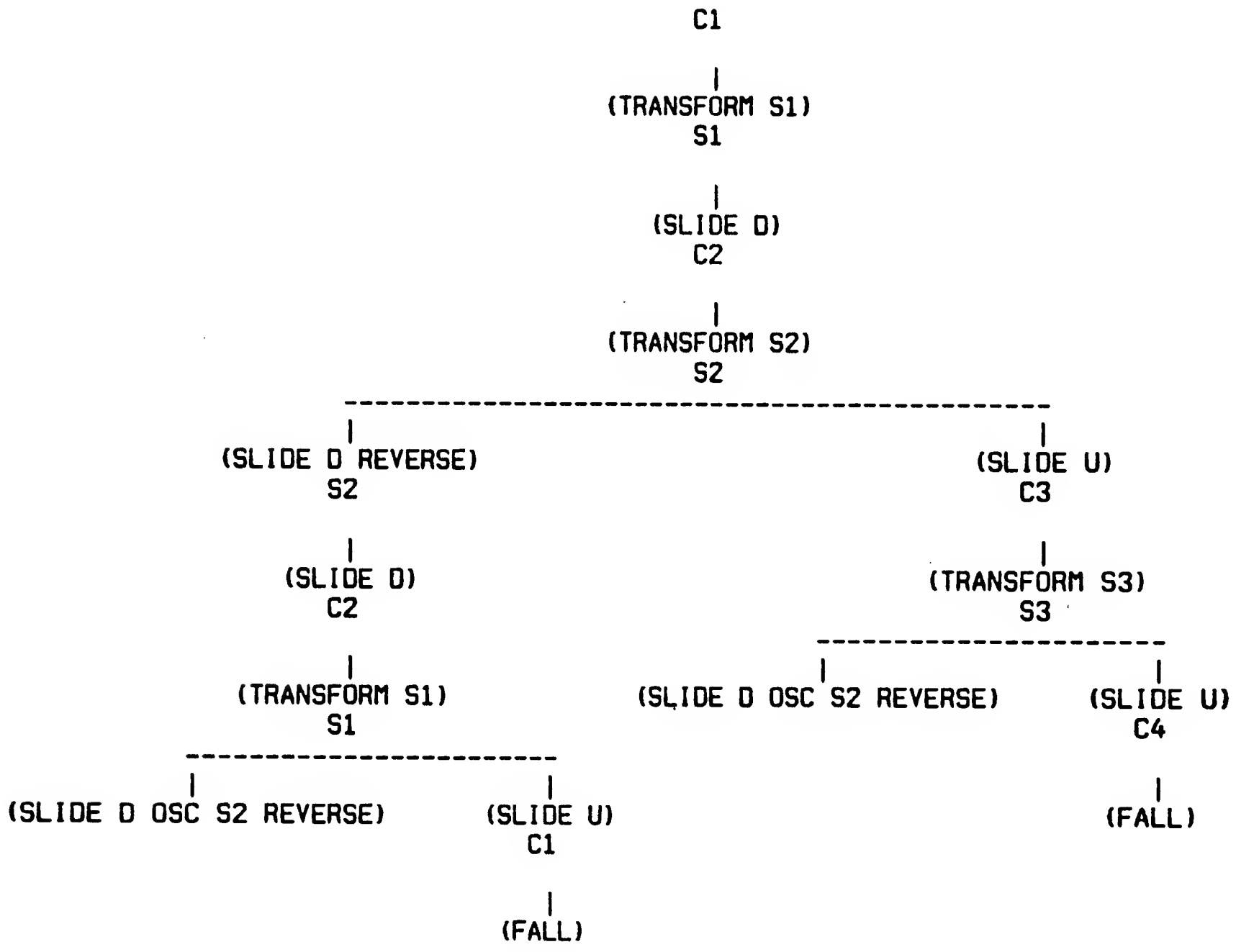
In this protocol we see the interplay of qualitative and quantitative knowledge. The possible path to reach X is described, obstructions in this path are recognized, equations are set up when the solution to the obstructions cannot be determined and then the solution to the equations are interpreted with respect to the qualitative ambiguity.

The first part of the protocol which involved identifying a possible path to reach X is *envistoning*. To envision is meant to intuitively imagine the event taking place. More formally, envisioning means generating a progression of scenes (which are encoded in some description) which describe what might or could happen. Features of this progression of scenes are used to obtain initial insights into the problem.

Envisioning describes (and so limits) the problem space formed when we 'ignore' the values of the variables (i.e. let the variables have arbitrary values). The problem had variables h_1, h_2, T and L which were required for the solution but the protocol (before the decision to calculate velocities) held true for a wide range of values for these variables. The reasoning depended on h_1

> 0, h_2 > 0, L > 0, 0 < T < 90 and the facts that all the curves were concave from the perspective of the object and that the curves were differentiable everywhere. All this information could be assumed from the diagram. Everything that was predicted by the envisioning was achievable for some values of the variables and every possible assignment of values to variables was described. Here we see what envisioning consists of: it considers only qualities (such as the sign of the slope of the surface) and then qualitatively describes all and only those possibilities which could possibly occur with those qualities being true.

The envisionment NEWTON would generate for this problem is:



The top of the tree indicates the initial point from which the object starts sliding. At each

subsequent node in the tree there is a descriptor (enclosed in parentheses) followed by the name of a corner or surface. The details of the descriptors will be explained in chapter 4. Each fork in the tree indicates that a number of possibilities occur there. The initial part of the tree can be read as: the object starts at corner C1, slides through segment S1, reaches corner C2, slides through segment S2, either slides back on segment S2 or reaches corner C3, and so forth.

The previous paragraphs describe what envisioning is; we will now examine the role it plays in problem solving. In the protocol we see that the initial approach to the problem is that of envisioning. One would expect that if the question was simpler it could be answered by envisioning alone. Questions such as "Will it reach the bottom?" are answerable from the envisionment. Envisioning fails to answer the problem when it predicts a number of possibilities. When the block was sliding up the hill, it could not be determined when or if it would start sliding back. It is in identifying these multiple possibility points that envisioner sets up specific problems. Even when further reasoning is required to resolve such a qualitative ambiguity, envisioning identifies those possibilities it must distinguish between. Although there was the problem of determining whether the block would slide back or not on the curve, the possibility of the block falling off had been eliminated by envisioning. In summary, envisioning gives local and very specific problems for further analysis and on a global scale envisioning provides an organization and plan to solve the entire problem. The trace of the possibilities through time provides the bases for such a plan.

In the protocol the solution method was to solve for each fork in the tree sequentially. This is one possible strategy. In NEWTON the question answering procedures could have realized that a positive solution to the second qualitative ambiguity would have also meant a positive solution for the first qualitative ambiguity.

In order to disambiguate between the possibilities occuring at each fork quantitative knowledge is used. Quantitative knowledge is represented in RALCMs (Restricted Access Local Consequent Methods). Each RALCM contains knowledge about a particular kind of situation. For example, one possible RALCM might be that collection of the trigonometric relationships which hold within a triangle.

On the following page a few of the possible RALCMs used to solve the current problem are given and then a scenario is outlined describing how these RALCMs can be used to solve the problem. In order to understand that scenario a brief oversimplified description of RALCMs will be given here. Chapter 5 will present everything in much more detail.

It is important to note that RALCMs are not procedures in the usual sense, instead they describe dependencies and assignments between variables. These dependencies are then examined by a general procedure IRALCM for the desired variable. RALCMs can complain back to the RALCM(s) that invoked them when they have a dependency which references the desired variable, but which also requires some other unknown variable. The first list of variables (designated by '?'s) of the RALCM body is its primary variables which must all be given when the RALCM is invoked. The primary variables describe the objects about which the RALCM is computing. The next list is the secondary variable a-list. Each of its elements describes a secondary variable. The first element of the pair is the atom which is used in the later equations. The second element of the pair is the global binding. This global binding must be expanded by substituting in for the primary variables.

The MASS-MOVEMENT RALCM knows about movements of objects on surfaces but is not concerned about the possibility that the objects may fall off the surfaces.

```
(DEFINE-RALCM  MASS-MOVEMENT  (?OBJECT ?SURFACE ?T1 ?T2)
                              ((A (ACCELERATION ?OBJECT)))

       (FCOND ((FEQ (TYPE ?SURFACE) 'STRAIGHT)
              ;if the surface is flat, try simple kinematics
              (FPROG ((THETA (ANGLE1 ?SURFACE)))
                     (RALCM RTRI (?SURFACE))
                     (RALCM KIN (?OBJECT ?SURFACE ?T1 ?T2))
                     (VSETQ A (* $%G (SIN THETA))) )))
       (RALCM ENERGY (?OBJECT ?SURFACE ?T1 ?T2)))
       ;energy will work for arbitrary shapes
```

```
(DEFINE-RALCM  ENERGY (?OBJECT ?SURFACE ?T1 ?T2)
                      ((VI (VELOCITY ?OBJECT ?T1))
                       (VF (VELOCITY ?OBJECT ?T2))
                       (H (HEIGHT ?SURFACE)))

        (EQUATION (= (+ (** VF 2) (- (** VI VI)) (* -2 $%G H)))))
        ;vf^2 - vi^2 = 2 g h

(DEFINE-RALCM  RTRI (?TRIANGLE)
                    ((H (HEIGHT ?TRIANGLE))
                     (L (BASE ?TRIANGLE))
                     (HYP (DISTANCE ?TRIANGLE))
                     (T1 (ANGLE1 ?TRIANGLE))
                     (T2 (ANGLE2 ?TRIANGLE)))

  (EQUATION (= HYP (SQRT (+ (** H 2) (** L 2)))))
  (EQUATION (= (SIN T1) (// H HYP)))
  (EQUATION (= (SIN T2) (// L HYP))) )

(DEFINE-RALCM  KIN (?OBJECT ?SURFACE ?T1 ?T2)
                   ((VF (VELOCITY ?OBJECT ?T2))
                    (VI (VELOCITY ?OBJECT ?T1))
                    (D (DISTANCE ?SURFACE))
                    (T (TIME ?T1 ?T2))
                    (A (ACCELERATION ?OBJECT)))

  (EQUATION (= VF (+ VI (* A T))))
  ;vf = vi + a t
  (EQUATION (= (** VF 2) (+ (** VI 2) (* 2 A D))))
  ;vf^2 = vi^2 + 2 a d
  (EQUATION (= D (+ (* VI T) (* .5 A T T)))))
  ;d = vi t + .5 a t^2
```

In order to get a better understanding of how RALCMs interact in the problem solving process we will artificially make h_1, h_2, T and L unknown and only provide their values when necessary. Later we will see what procedures must exist between the envisioning and the quantitative knowledge of RALCMs. For the moment let us call the knowledge not directly involved with RALCMs qualitative knowledge. In this scenario the RALCMs will fail back to the qualitative knowledge.

Envisioning determines that there is never a possibility that the object will fly off. Also it isolates the problem into the two necessary subproblems: first it must be determined whether the object reaches S3 and then whether it slides back on S3.

First the velocity at the end of S1 must be found. To do this MASS-MOVEMENT must be

invoked:

(MASS-MOVEMENT (B1 S1 TIME1 TIME2))

Before MASS-MOVEMENT can be invoked variables must be assigned values and meanings:

(VELOCITY B1 TIME1) - known

(VELOCITY B1 TIME2) - unknown

'Unknown' indicates that this is the value MASS-MOVEMENT is required to find. When MASS-MOVEMENT is invoked IRALCM attempts to find a value for this variable. By searching through the RALCM it is found that there are two places in MASS-MOVEMENT in which a possible assignment to VF take place. IRALCM quickly discovers that the assignment in the FCOND is impossible to reach as (FEQ (TYPE ?SURFACE) 'STRAIGHT) is unsatisfiable. The only alternative is the ENERGY RALCM. When ENERGY is invoked it complains back that it was given insufficient information, namely it was not provided with the HEIGHT. Every possible attempt to achieve a value for VF has now failed, the alternative is to examine the reasons for these failures and attempt to eliminate them. The complaint blocking the only path to VF is that HEIGHT is unknown. There are no accessible references to HEIGHT in the RALCM so it fails back to the qualitative knowledge complaining that it was not provided with height of the surface. The qualitative knowledge must now find the height or find the problem impossible to solve. HEIGHT is given. IRALCM can now complete its examination of ENERGY and return a value for the unknown VF which MASS-MOVEMENT subsequently returns to its caller. This value is remembered and the segment S2 is examined in much the same way using the VF on S1 as VI on S2:

(MASS-MOVEMENT (B1 S1 TIME2 TIME3))

Note that ENERGY returns an "impossible" result if the equations result in an imaginary solution, thus indicating that the object cannot traverse S2.

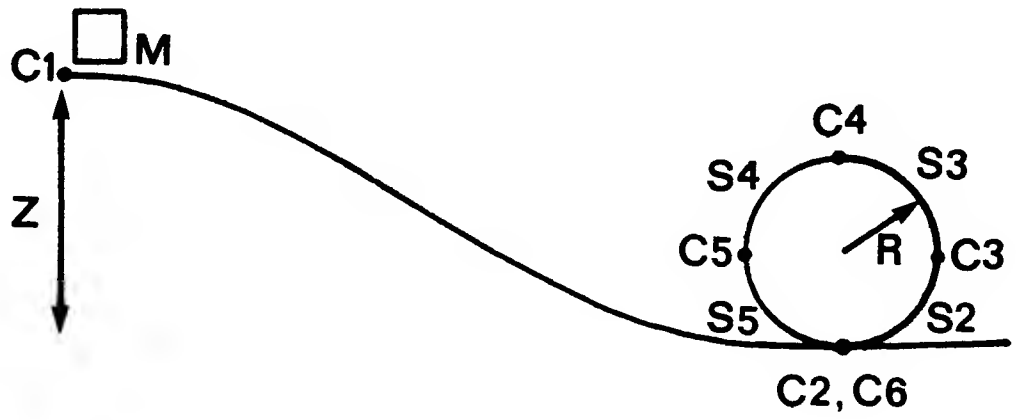
On S3 IRALCM has two possible paths to a solution. If ENERGY is tried it fails because HEIGHT is unknown. Since (FEQ (TYPE ?SURFACE) 'STRAIGHT) is satisfied KIN can be tried immediately for a solution. KIN complains back that it must know the value for either D or T

before it can proceed. Again every path to VF is blocked and these complaints must be examined to see if they can be satisfied. The two alternative complaints that block the path for finding VF are (1) find HEIGHT and (2) find D or T. There are no other references to T in the RALCM so T cannot be achieved in MASS-MOVEMENT. The two variables HEIGHT and D can be found by the RALCM RTRI. RTRI is then invoked on segment S3. Of course there is not enough information to solve for the variables so complaints are generated back up to the qualitative knowledge. The qualitative knowledge sets values for T1 (angle of surface) and L (base length of surface) in the instantiation of RTRI on S3 and IRALCM proceeds. Now RTRI returns with values for both D and HEIGHT. IRALCM has a choice whether to restart KIN or ENERGY to solve the problem. Depending on whether this results in an "impossible" solution or a particular value for VF the question of whether C4 is reachable has been answered.

The solution outlined here is not necessarily the best one. There a number of other solutions NEWTON would generate first. If the envisioner was told explicitly that X was higher than C1 it would have eliminated the possibility of sliding back on S3 or S2. Furthermore, if h_1 and h_2 were given as numbers the qualitative comparison between the heights would have been generated. Or, a global strategy could have immediately decided to just attempt to compute the height between X and C1.

2.3 A Loop-the-Loop Problem

This problem is intended to further illustrate envisioning and also to elucidate the kind of reasoning which must take place between the envisioning and the quantitative analysis.



A small block of mass m starts from rest and slides along a frictionless loop-the-loop as shown in the figure. What should the minimal initial height z be so that the block successfully completes the loop-the-loop?

"The object will slide down the incline into the loop, it may slide back on the way up, fall off before it reaches the top or it may fall off after the top. The crucial point is the top where the gravitational force pulling down on the object must be less than the centripetal force due to its circular motion if the block is not to fall off. For the object to just stay on the surface these forces must be equal:

$$m v^2 / r = m g$$

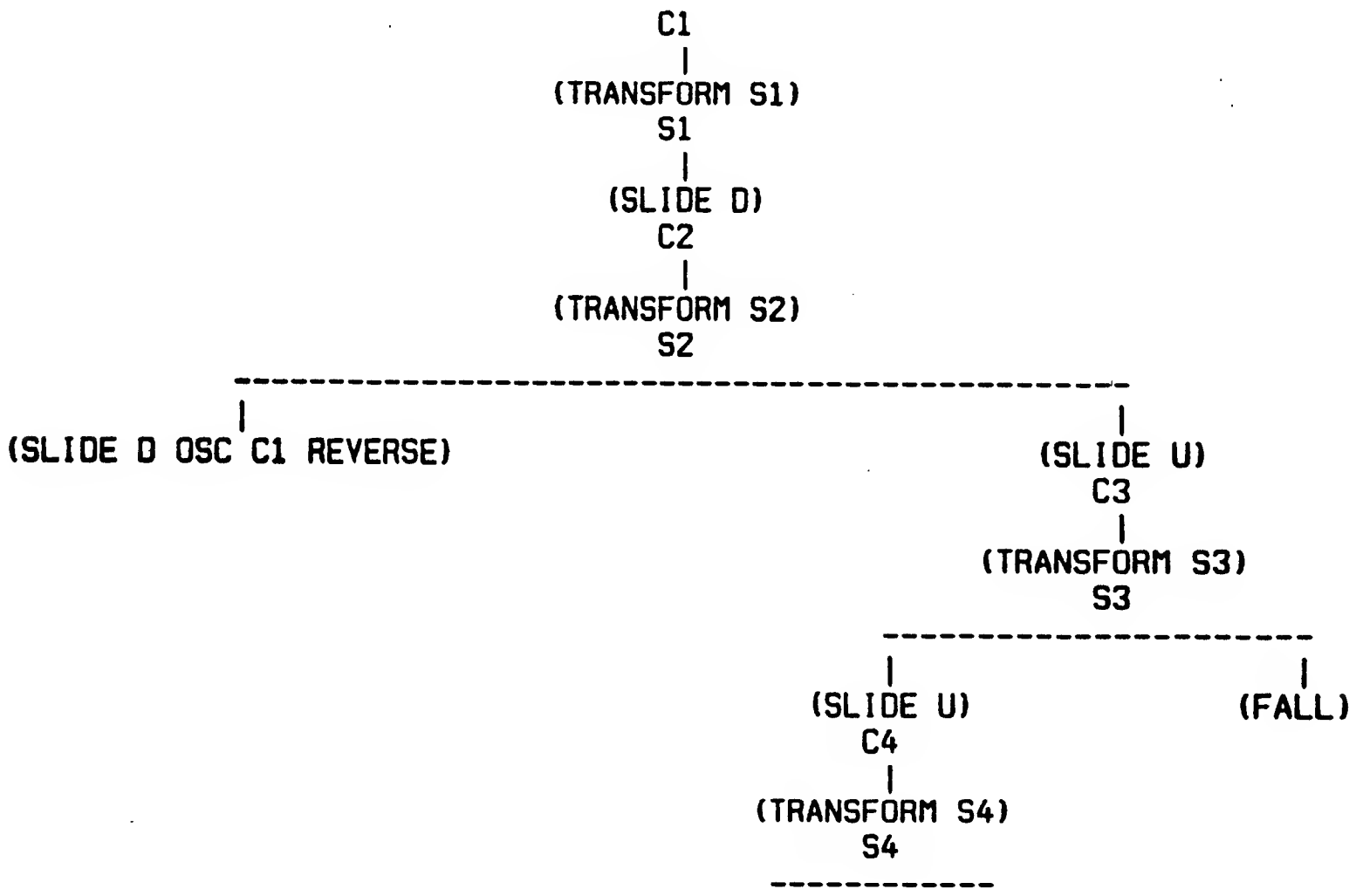
In order to calculate v, conservation of energy could be used.

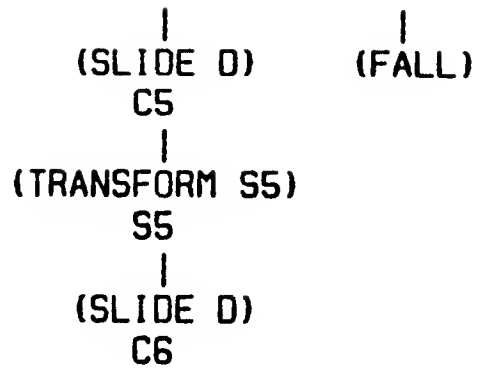
$$1 / 2 m v^2 = m g (z - 2 r)$$

Solving for z:

$$z = 5/2 r."$$

The envisionment for this problem is:





The envisionment for this problem is much more difficult. For NEWTON to envision this problem correctly it had to know that objects moving underneath can surfaces fall off but, that if the surface was convex with respect to the object it might stay on.

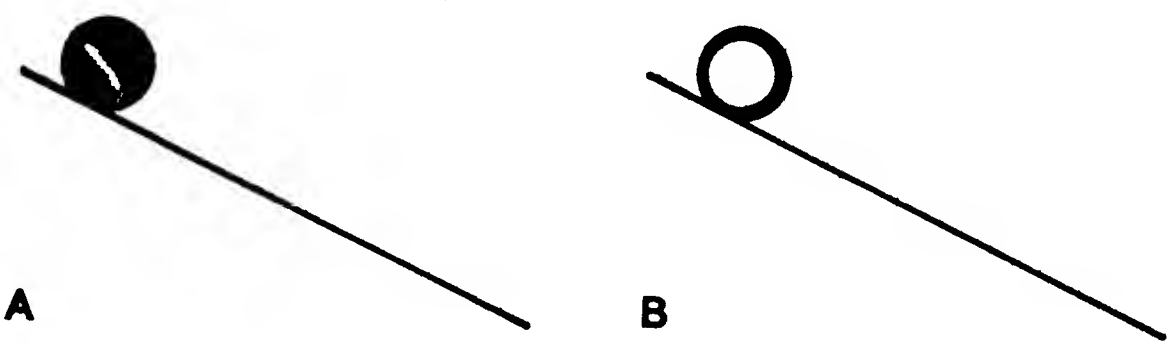
In this problem we see an example of a global reasoning strategy. Using the local strategy of the previous chapter we would have had to solve three sequential problems: sliding back on S2, falling off on S3 and falling off on S4. In fact the problem could be solved this way; however, the global strategy realized that if the object did not fall off on S3 and reached C4, the other two difficulties would automatically be resolved. This is a very powerful local vs. global principle: find those local problems whose positive solution will automatically determine the desired solutions for the other local problems. Often the desired local point to select is the last one, but that is not always the case. For example, there might be no global procedure to handle the problem; all global strategies presuppose something about the local situations they span, the usual energy law applying in this loop-the-loop problem would be inapplicable if there were friction on S2.

There are many different kinds of question types. These will be described in much more detail in chapter 4. One fundamental distinction lies between boundary conditions and final result. In the sliding problem the boundary conditions were specified and the question was about the result. In this loop-the-loop problem the result is specified and the boundary conditions which lead to that result need to be determined.

This loop-the-loop problem illustrates the distinction between variables, parameters and values. An actual solution will require a numerical value and if the equation for the result contains an unknown variable it is not in general an acceptable solution. In some circumstances the problem

can explicitly give a variable for which the value is not necessarily known yet is allowed to appear in the final solution. Such a variable is called a parameter of the problem. In the loop-the-loop problem r was a parameter and the final solution for h is stated in terms of it. A parameter is then a variable which can be treated as if it were known, yet have no numerical value. Usually, as in the loop-the-loop problem, the parameters are explicitly given. Some problems require that the problem solver choose the appropriate set of parameters. In any case NEWTON will try to find a solution employing a minimum number of parameters. If the parameters were explicitly given it will try to choose the minimum possible, if it has to choose its own parameters it will try to find a minimal set.

2.4 A Comparison Problem



Two spheres of equal mass roll down an inclined plane without slipping. Both spheres are of equal mass and radii, but sphere A has a uniform density and sphere B is hollow. Which one will reach the bottom first?

"Both spheres roll down the plane. If one of the spheres reached the bottom faster it would have reached any point in between faster so all we have to do is compare velocities at any point. The velocity at any point is directly proportional to kinetic energy of translation. The change in potential energy is proportional to the height through which the sphere moved. The only other place this energy was used was in the kinetic energy of rotation. The kinetic energy of rotation varies directly with the moment of inertia of the object and its angular velocity. Since there is no slipping, the angular velocity is directly proportional the translational velocity. So the higher the moment of inertia is the higher the energy of rotation is and the less energy is in the translational motion. Hence, given that the mass and radii are fixed the sphere with the lower moment of inertia

will reach the bottom faster. The answer to the problem is then sphere A."

This problem and its solution are radically different than the previous two problems. First, envisioning played only a minor role and second, apparently no quantitative knowledge was used. The problem might have been solvable using quantitative knowledge by substituting in for the variables, or just solving for the time in general and then comparing the resulting equations. Or, substituting numbers and comparing might also work but one has to then know that the quantities vary monotonically with each other.

Such a quantitative way to solve this problem seems very poor and the quantitative strategy would break down if the spheres were rolling down an unspecified surface (but guaranteed not to fall off). For this more complex problem the above argument still holds almost exactly but the quantitative strategy would completely fail.

The kind of knowledge used to solve this problem was about how which quantities depended on what and how. Such knowledge about the functional dependencies between variables makes certain problems much easier to solve, and even allows one to solve problems which previously were unsolvable. The ability to manipulate functional dependencies allows the expression of many more kinds of knowledge.

2.5 Summary

Many of the different kinds of knowledge involved in solving physics problems have been discussed in this chapter. The following chapters will develop these kinds of knowledge in much greater detail along with other kinds of knowledge. There remains many kinds of knowledge that will not be considered here. For example, the problem of learning this knowledge or just using this knowledge in new situations which violate some of the prerequisites for applicability. The physics student is a model builder, learning a model and spending a long time discovering exactly under what circumstances the model is valid and under which it is not. There is the ability to handle metaphor and analogy; how can we use this knowledge about mechanics to understand current flow in electronics? More specifically there is no ability to quantify over structure, there is only an ability

to quantify over something that is parameterizable as a variable. NEWTON neither qualitatively or quantitatively understands the calculus of variations. In fact NEWTON does not even know enough calculus to realize that integration is a quantitative operator. NEWTON does not represent the kind of knowledge that would make it possible to construct a program like itself. Although NEWTON may employ a large amount of common sense reasoning ability, it understands nothing of the common sense reasoning the author used to develop it or that Galileo <Galileo, 60> used to break away from the shackles of Aristotelian physics .

From this chapter it should be even clearer that the distinction between qualitative and quantitative is very fuzzy. There are many kinds of qualitative knowledge such as envisioning and question answering, and there are many kinds of quantitative knowledge such as that contained in our RALCM and some aspects of comparison. The position of question answering and comparison is not very clear. For purposes of discussion we will use qualitative and quantitative to refer to the kinds of knowledge listed above collectively. In subsequent chapters these lists will become longer and fuzzier.

3.0 ENVISIONING

3.1 What it is

The most basic and primitive knowledge about physics is envisioning. Intuitively, envisioning is imagining an event taking place. For NEWTON envisioning means generating a progression of scenes encoded in a symbolic description which describe what could happen. Both in NEWTON and in people the envisionment of the event is the first step in the understanding of the problem.

In fact, envisioning is necessary to understand the event at all. To understand that a pencil might roll off a table requires the ability to envision that event without actually seeing it happen. We see, then, that envisioning is pre-physics knowledge and its presence is independent of the goal of solving mechanics problems. Envisioning is, however, the fundamental building block for understanding mechanics.

The process of studying mechanics involves, in a large part, the building of connections between the envisioning knowledge and the mathematical knowledge. The learning of mechanics will involve acquiring a better understanding of the world the envisioning was about and consequently that education would refine the envisioner. Any refinement that does take place is minor and fits within the original framework.

The resulting envisionment is used as a basis for further reasoning. This envisionment involves the two most fundamental ideas of problem solving. First, the envisionment (with its connections with quantitative knowledge) produces a description and representation of the problem appropriate for further analysis. For example, envisioning recognizes the geometry of the event. Second, the envisioning gives rise to a plan to analyze the problem which is linear in time. Simple quantitative techniques for following such plans exist and further reasoning can be conveniently done on such linear plans to find better nonlinear plans.

3.2 How Envisioning is Done

The envisioner consists of three sections: local experts, global experts and control structure. The local and, to some extent, the global experts are domain dependent and capture knowledge particular to the given domain. The control structure is essentially domain independent and interacts with the local and global experts through fixed and predetermined interfaces.

The scene being envisioned must be encoded in some description. This description divides the scene into physically local sections. The exact nature of this description will be left until after the local experts have been presented. There are two kinds of local experts: feature experts and action experts. The feature experts take into account local features and deduce subsequent actions. These actions take place over time. The action experts take a given local area and an action and deduce which next local area should be examined. All this interaction is monitored by the control structure which also keeps a record of all the interactions. Because the same feature expert can generate many actions, the control structure has to decide which action to try next. The resulting record is a tree with forks appearing where multiple actions are possible. This tree is subsequently pruned by the global experts.

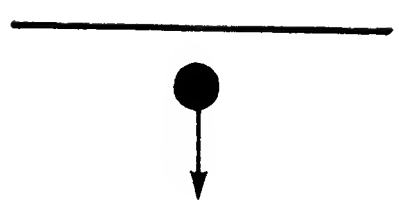
There are six basic kinds of actions that arise in our mini-world. Each action must look at the description of the environment to determine which local area should be examined next. There is the possibility of no action: STOP. On a surface movement along it can occur: SLIDE. Sliding through a point is considered different than sliding over a segment: TRANSFORM. An object can fall off the surface: FALL. It can also fly off: FLY. At a corner the object can collide with another surface: COLLIDE. Some of these actions may have an argument associated with them to give particular details of the action.

Ignoring, for the moment, the velocity, the following five possibilities can arise at a microscopic level on a surface. At the microscopic level all curves are linear.

Object is off the surface:

-> FALL

Object is below the surface:



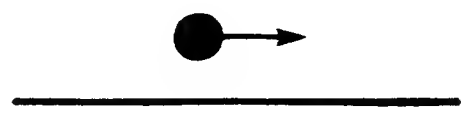
-> FALL

Object is on a horizontal surface:

->STOP

Object is on an incline:

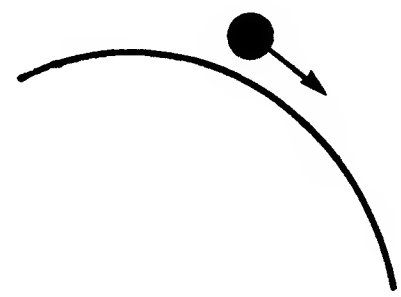
If the object is moving with a nonzero along a horizontal surface:



->SLIDE

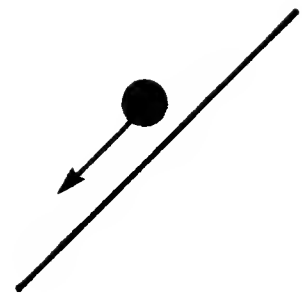
The analysis so far has only examined a microscopic section of any curve, we would like to employ these experts on larger sections. Taking into account larger sections of the curve introduces the feature concavity. With concavity taken into account two other possibilities arise.

If the object is moving on top of a surface which is concave away from the motion, the object might fly off:



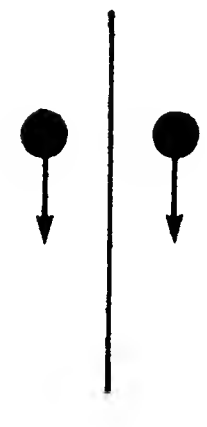
->FLY

If the object is moving underneath a surface which is concave into its motion the object might remain in contact with the surface:



->(SLIDE D)

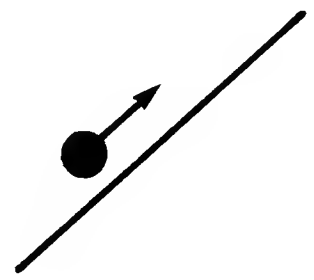
Object is on a vertical surface:



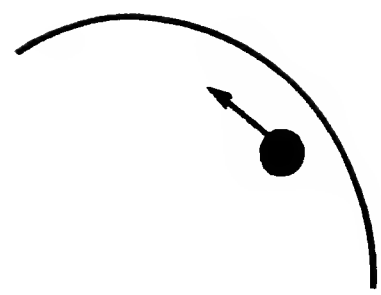
->FALL

With zero initial velocity there is never any doubt what happens. If the object had, however, an initial velocity there could be another possible action.

The object is moving up against gravity:

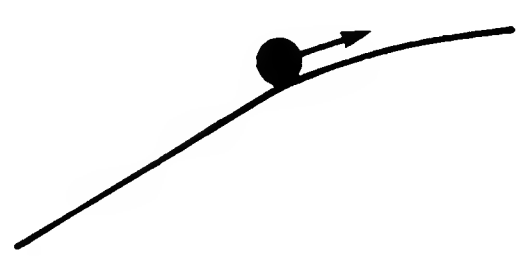


->(SLIDE U)



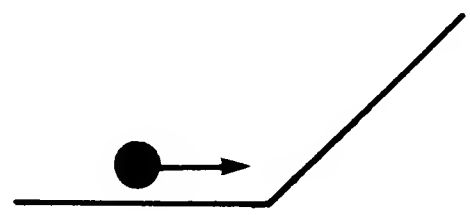
->(SLIDE U)

At corners, which are considered as points not on a segment, three possibilities can occur. If the curve has a continuous derivative at the corner the object remains on at the corner:



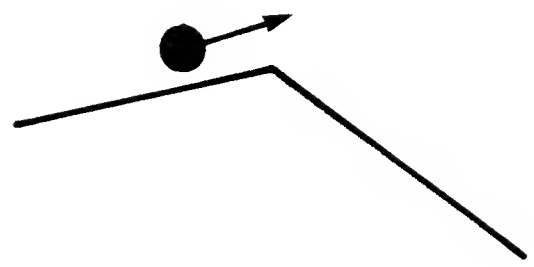
->TRANSFORM

The corner could be sharp, folding back into the path of the object:



->COLLIDE

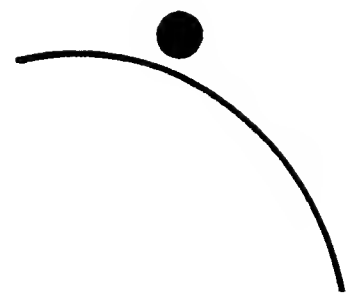
The corner could be sharp, folding away from the path of the object:



->FLY

These rules must be combined to perform a complete envisioning. Although concavity has been recognized there still is no way to combine the rules to produce a complete envisionment. The answer lies in the description of the environment. The rules already imply a great deal about how the environment should be described. The crucial quantities are the first and second derivatives of the curve. One of the most powerful principles of envisioning is that in between the singularities and zeros of a quantity affecting an action, the effect varies continuously. More simply, the singularities and zeros are the points at which violations of the rules are most likely to occur. This important principle is embodied in the rules for vertical, horizontal and inclined surfaces. All inclines are equivalent as far as acceleration is concerned. If the environment is described in terms of the desired zeros and singularities the rules can be used. We will formalize the notion of a segment and corner. Points on the curve at which singularities and zeros occur will be called corners and the sections of the curve between them, segments.

We still do not have all the details to be able to envision movement on a segment. One simple minded way is to apply the rules, observing the resulting changes and applying induction. A second principle of envisioning saves us from this undesirable option. If the actions indicated by the rules do not change the description of environment which the rules rely on, then the analysis for every point on the curve will be the same. With one exception, which we will now rectify, this principle can be applied to all the rules outlined so far.

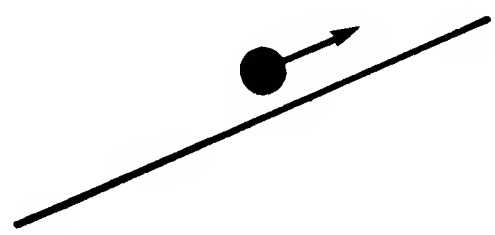


If an object were placed on the indicated surface with a zero initial velocity, the object might fly off. This possibility would not be detected by any of the rules. Rules that take into account the velocity of the object should take into account both actual and potential velocity. This observation leads to the introduction of a single rule to handle the above case.

The state of the object can be encoded in a triple indicating whether the object is on or off the surface, above or below the surface and the sign of its velocity:

{{on,off},{above,below},{velocity-u,velocity-d,velocity-z}}

An object sliding up an inclined surface would be described as:



{on,above,velocity-u}

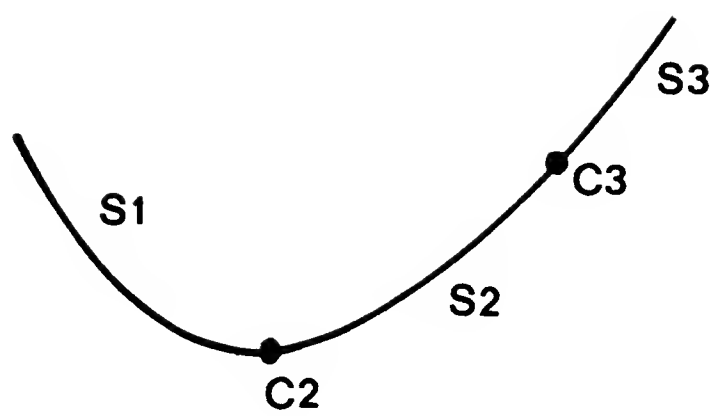
A segment is described in terms of its first and second derivative.

(SEGMENT name {incline,horizontal,vertical},{minus,zero,plus})

A corner is described by the change of slope occuring there.

(CORNER name {minus,zero,plus})

The environment of the sliding block problem of chapter 2 could be described:



(SEGMENT S1 INCLINE PLUS)

(CORNER C2 PLUS)

(SEGMENT S2 INCLINE PLUS)

(CORNER C3 ZERO)

(SEGMENT S3 INCLINE ZERO)

The rules can be conveniently expressed as productions. The left side of the production is in terms of the features of the environment and the right side indicates the resulting possible actions. All applicable productions are applied to determine the possible actions. By the second principle of envisioning the actions of the left hand side should never produce a change in the features so that each production need only be examined once.

The rules for a segment are:

(1) off → fall

(2) below → fall

(3) horizontal ∧ on ∧ velocity-? → slide-?

(4) incline ∧ above ∧ on → slide-d

(5) vertical ∧ on → fall

(6) velocity-u ∧ incline ∧ above ∧ on → slide-u

(7) convex ∧ on ∧ above → fly

(8) concave ∧ on ∧ below ∧ velocity-? → slide-?

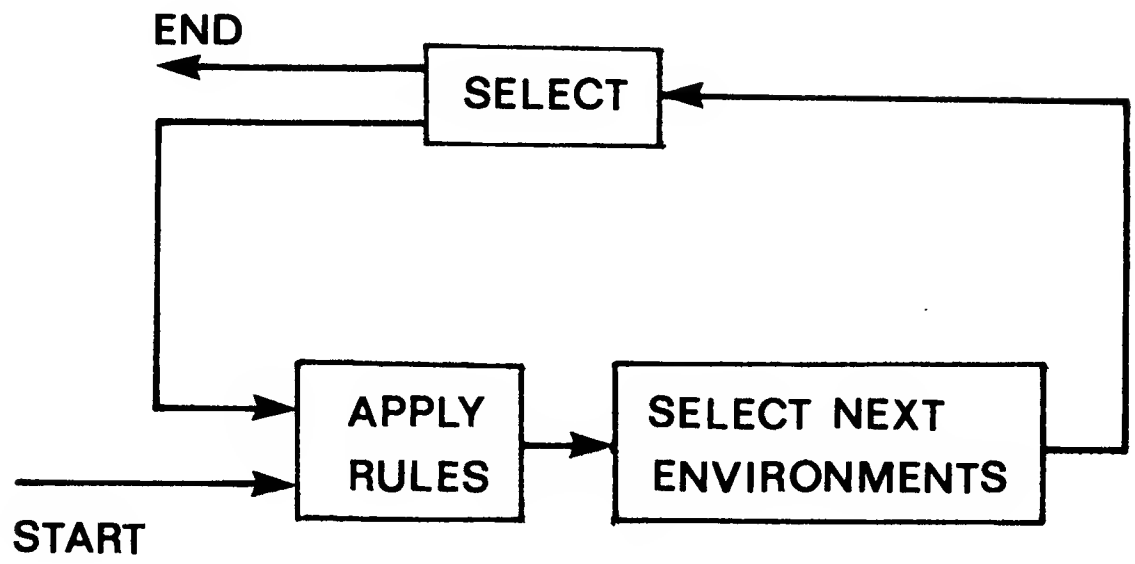
The rules for a corner are:

(9) zero → ϵ

(10) plus → fall

(11) minus → collide

The control structure for the envisioning is very simple:



The resultant trace of the interaction is the envisionment and takes the form of a tree.

As an example the envisioning for the sliding block problem will be examined in detail. The object is placed at the start of segment S1 with zero initial velocity. The initial state is then:

{on,above,velocity-z}

(SEGMENT S1 INCLINE PLUS)

Production (4) applies, giving slide-d. This single possibility results in the movement to corner C2.

{on,above,velocity-d}

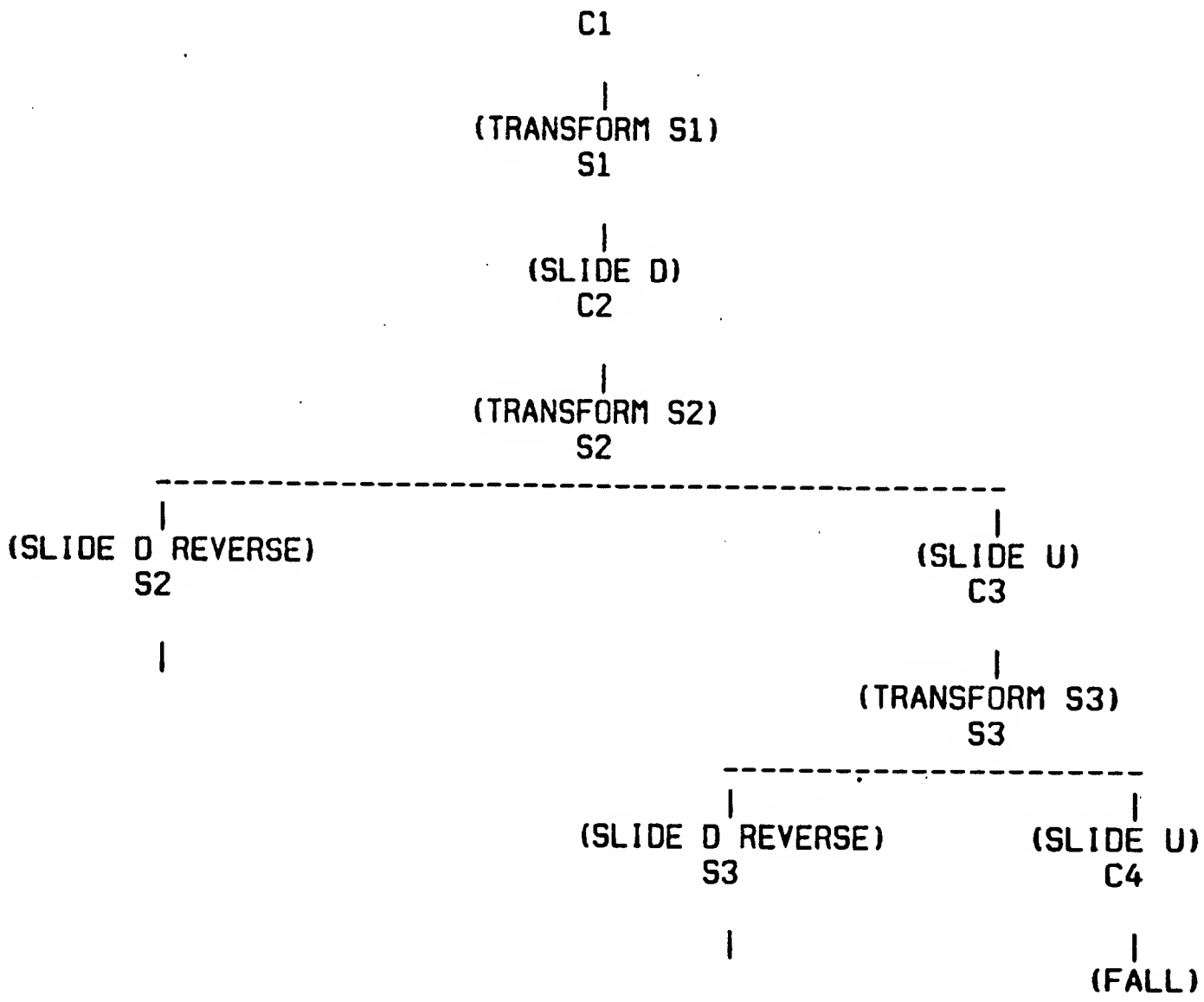
(CORNER C2 ZERO)

Only production (9) for the corner applies, so the next segment is selected. Since a downward velocity from S1 and across C2 implies an upward velocity on S3, that feature must be changed.

{on,above,velocity-u}

(SEGMENT S3 INCLINE PLUS)

Productions (4) and (6) apply. This results in two possibilities **slide-d** and **slide-u**. These two possibilities result in the selection of C2 or C3. If the C3 option is chosen an identical analysis takes place on S3. This short scenario has resulted in the generation of a tree:



3.3 Global Experts

Global experts observe features of the tree as it is being generated and prune it. The necessity for the most important global expert arose in the example envisionment of the previous section. The envisioner goes into an infinite loop when oscillation is present. The only other global expert that NEWTON uses is a qualitative conservation of energy expert. NEWTON uses the relative heights to eliminate certain possibilities in the tree. NEWTON can be told explicitly about

the global features of relative heights and the envisioner will prune the envisionment according to the implications of conservation of energy.

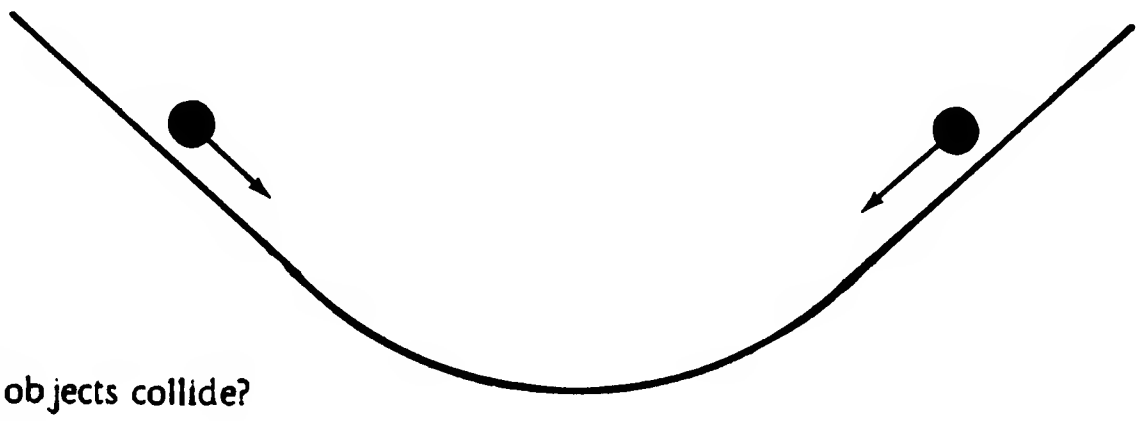
Oscillation occurs when a state is repeated. The state of an object is given by its velocity and position. There is, however, a more elegant and generalizable characterization of oscillation. Consider those points for which the velocity is momentarily zero. We will call these the stagnation points. A possible path can have either zero, one, two or more stagnation points. If the path has no stagnation points the object has been given an initial velocity and the velocity never went down to zero. A path with a zero initial velocity and one stagnation point has the same analysis. A nonzero initial velocity and one stagnation means that one velocity reversal took place and the object eventually left the surface. In every case where there are two stagnation points oscillation takes place. If the initial velocity was zero one of these stagnation points is the initial point on the segment, otherwise the object was given an initial velocity on an oscillatory path. Oscillation can then be simply recognized by keeping track of how many stagnation points have been encountered. When oscillation is detected the action (OSC point) is inserted in the tree indicating the object would oscillate back to the specified point. The envisioner then stops envisioning that particular path.

NEWTON has a procedure to handle inferences about partial orders. The relative heights of the points on the curve form such a partial order. The envisioner has a simple theory of conservation of energy which uses this partial order to determine if velocity reversal could take place. For NEWTON to be able to make any useful inferences it must be told something about the relative heights in the diagram. Each path is marked with the highest points in the partial order that it has reached. If local experts deduce that a stagnation point is possible on a segment then the heights of the endpoint is compared against the heights of that path so far. If comparison is possible the correct choice can be made.

3.4 The Role of Envisioning

The tree generated by the envisionment is used to guide further analysis. The forks in the tree are points at which the envisioner could not disambiguate the possibilities. As we shall see in the next chapter this same tree can be used in many different ways. For example, it can be used to determine whether a given question is absurd or answerable.

Envisioning is a fundamental piece of knowledge required to be able to reason about mechanics. The envisioner presented in this chapter can be generalized in many ways; however, its structure would have to be fundamentally modified to be able to do the envisioning required for general mechanics. The organization of the envisioner makes the fundamental simplifying assumption that all the interesting interaction taking place at any given time occurs in one small local area. It has only one center of attention. This leads to difficulties that have to be resolved before general mechanics envisioning can be done. The first difficulty is that two physically unrelated centers of attention may interact in the future. NEWTON is totally incapable of envisioning a collision involving two objects.



Where will the objects collide?

This requires the envisioner to understand relative times.

The other difficult situation is two interacting centers of attention.

Will the object make the loop-the-loop?

Clearly the block is too large. NEWTON cannot envision anything other than a point mass. Envisioning an object with a volume requires paying attention to all the boundaries of that object. The same problem arises in understanding a lever or a string.

The assumptions that the object really is a point mass and that there is only one center of attention gives a good basis to build the envisioner upon. There are usually only a small number of centers of attention in any problem. The handling of nonlocal physical interactions required different features descriptions and actions, but this sophisticated interaction would still result in a tree of possibilities. In any case the matter requires more serious research, especially into descriptions of objects and actions.

There remain some interesting generalizations to the envisioner that have not yet been discussed. Friction fits within the simple envisionment paradigm. Unfortunately, the introduction of friction requires modification of all the global experts. Damped oscillation becomes a possibility. Collisions with surfaces can easily be added. If the issues of boundary interactions could be ignored rolling balls or other larger objects could be handled. There remain many other possibilities. However, NEWTON can currently only do the kind of envisioning that has been outlined in this chapter.

4.0 QUESTION ANSWERING

4.1 Examining the Question

The mechanics problem as presented to NEWTON consists of an event and a question about that event. NEWTON always does at least a partial envisionment of the event regardless of the specific question asked. Envisionment is necessary to understand the problem at all. The envisionment sets up an information structure describing the event. Only after this structure has been built is the specific question about the event examined. For example the envisionment references the appropriate fragments of quantitative knowledge so that when a quantitative question is asked a search for information about it can be limited to only those fragments of quantitative knowledge in the structure. There are many different kinds of questions and each different kind uses the information structure generated by the envisionment in a different way. In this chapter some of these different question types will be examined and strategies outlined to handle them.

Questions can be divided among many different dimensions. The most fundamental division exists between questions about final effects derived from given boundary conditions and questions about boundary conditions given the final effect. A question can be quantitative "What is the velocity?" or qualitative "Will the block fall?" Some questions quantify over structure: "Where will the block fall off?" Other questions request information about global quantities: "What was the average velocity of the block?"

4.2 Common Sense

NEWTON can understand the same problem at many different levels. So far we have seen three of the levels: envisioning, question answering and quantitative knowledge. The general strategy is to attempt to solve the problem at the simplest possible level and use the problems the simpler analysis gets into to guide the more sophisticated further analysis. If the envisionment describes an event different than specified in the question the question must be absurd. If the QAs, which observe more global features, notice that the envisionment does not have the appropriate features the problem is also absurd. It is this combination of envisioning and question answering

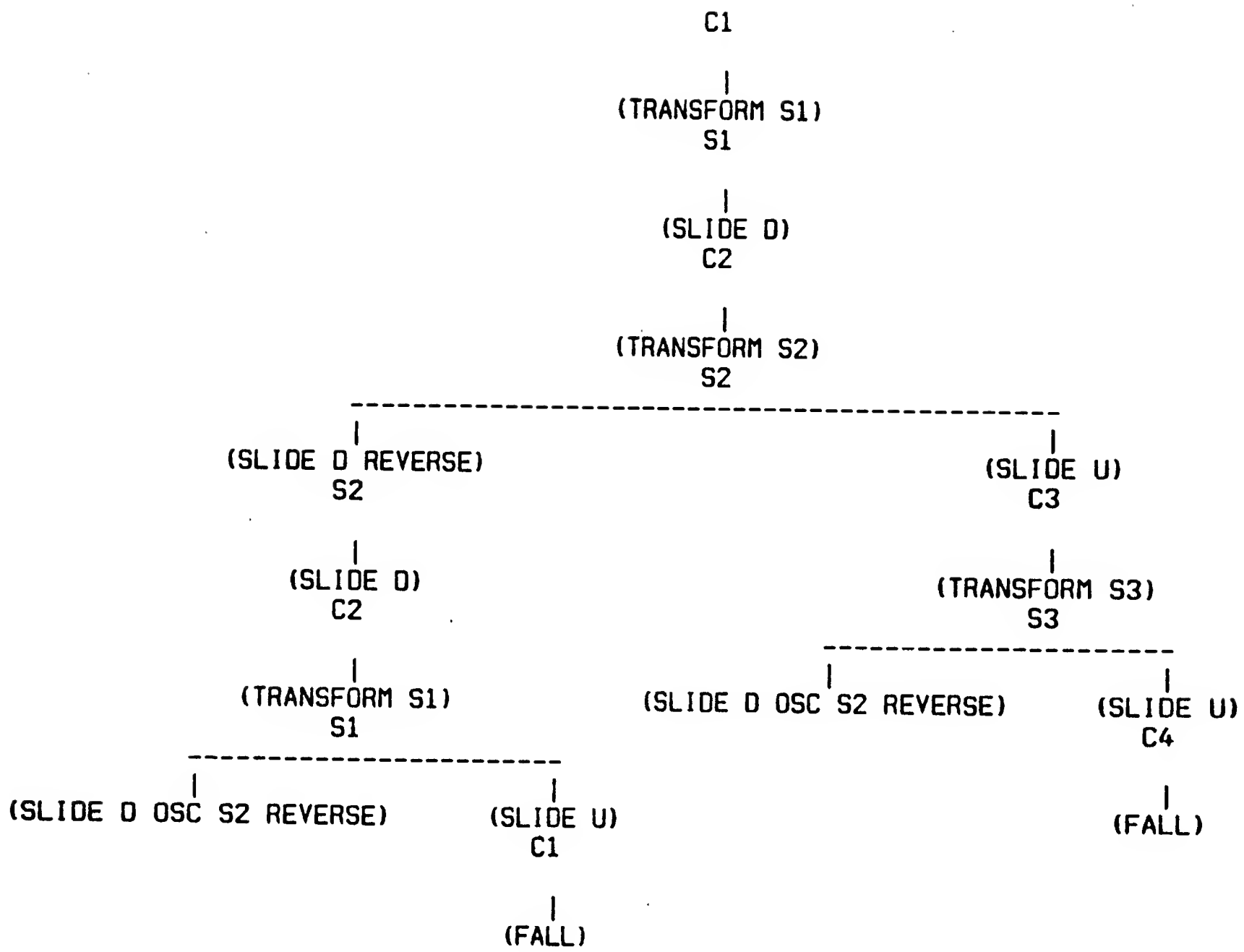
knowledge that gives NEWTON much of its common sense. These first two levels of knowledge are extremely general and versatile, being able to accurately identify exactly those points at which further problem solving has to be done. (Often the quantitative analysis will be incapable of dealing with the identified difficulty). Simple questions are dealt with simply and absurd question are recognized immediatedly.

One fundamental step is to realize that any question has many implicitly unstated preconditions. These preconditions are less complex than the final problem and their examination even if proven positive provides more information for further analysis. If negative, the final problem cannot be solved (it is absurd) since its implicit preconditions are violated. For example, given the qualitative question "Does the object reach X?" it should be ascertained whether there is any path to X. A similar analysis applies to quantitative questions. The question "What is the velocity of E at X?" implies that the simpler qualitative question that the point X is reachable at all.

4.3 Simple Descriptive Questions

The simplest kind of question is just a request for a description of the event given the initial conditions. We will use the strategies to answer this question to illustrate the basic pieces of knowledge that will have to be used for more sophisticated question types.

The envisionment produces a description of the event containing forks or qualitative ambiguities. Consider again the envisionment for the sliding block problem of section 2.2.

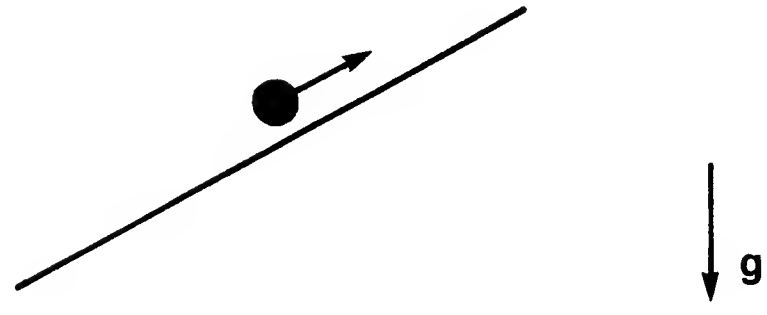


To determine what happens, the first fork is examined and disambiguated.. This procedure is repeated on the next fork. The result is a (partially qualitative) description of the event. If the left branch is taken at the second fork event would be an oscillation between a point on S1(C1) and a point on S2. If the right branch is taken the object subsequently leaves the surface.

The envisioner of the previous chapter can only generate five different kinds of qualitative ambiguity. (For the purposes of qualitative ambiguity the action OSC is really a loop back into the tree and hence really is a SLIDE action.) The possible forks, their meaning and an example of each are:

SLIDE-SLIDE:

The object is moving and a deceleration is being applied which may result in a velocity reversal.

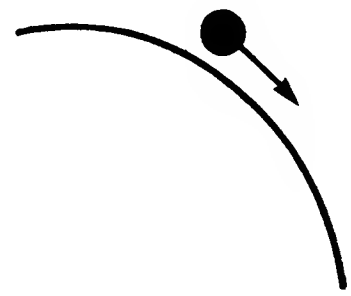


FALL-SLIDE:

The object may be being accelerated or decelerated, however, it may fall off from moving underneath the surface. Velocity reversal is not an issue here since if the object ever achieved zero velocity underneath the surface it would instantaneously fall off the surface.

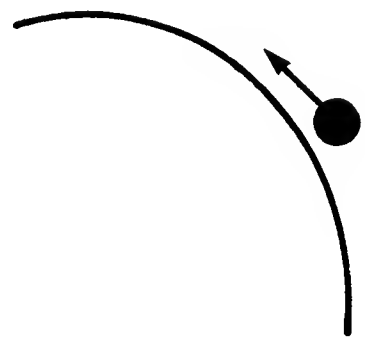
FLY-SLIDE:

The object is being accelerated to the point where its velocity may eventually be great enough to fly off the surface.



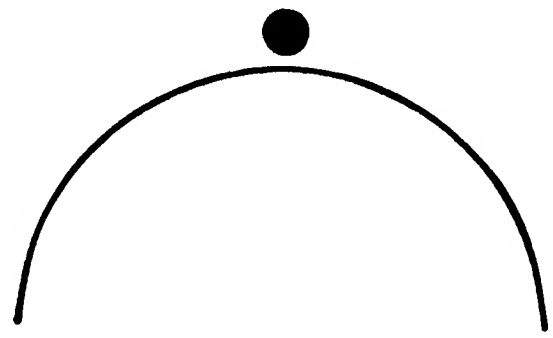
FLY-SLIDE-SLIDE:

The object is being decelerated but the velocity in either direction may be sufficient for it to fly off the surface.



TRANSFORM-TRANSFORM:

This is the only qualitative ambiguity which is considered at a point. The previous ambiguities may occur at a point but they are considered in the context of the surrounding segment and not at the point itself. This qualitative ambiguity is unsolvable and external advice is necessary. The object has reached a point of unstable equilibrium.



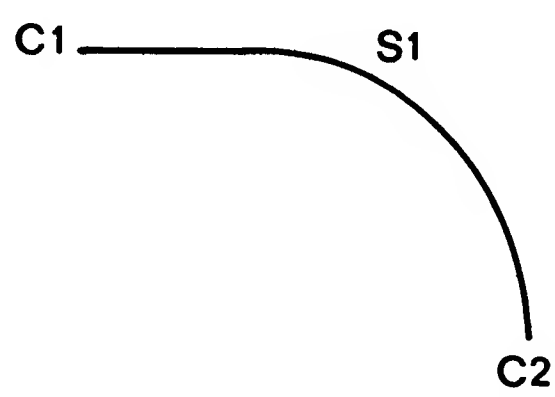
These are the fundamental kinds of qualitative ambiguity. In later chapters others will be introduced as the envisioner is made more sophisticated.

For each kind of qualitative ambiguity there exists a collection of RALCMs which can be used to resolve that ambiguity. Many of the RALCMs for different ambiguities reference each other. We have already seen one such collection of RALCMs for SLIDE-SLIDE: MASS-MOVEMENT. To obtain a simple description of the event the envisionment is performed to obtain a tree and the quantitative knowledge is applied to resolve the ambiguities at the forks.

This is all that has to be done to obtain a simple of the event, but often other features are desired regardless of whether a fork occurred. For instance, the velocity at a particular point may

be desired. Even if a fork occurs one might want to know where the object falls and not just whether it falls off. In most cases the precise value of the desired parameter can easily be calculated by the RALCMs resolving that particular ambiguity. The parameter need not necessarily be determined in the process of the resolving the ambiguity, however, the collection of RALCMs used to resolve the ambiguity must provide sufficient information about the desired parameter so that it can be determined.

Unfortunately for the purposes of obtaining exact values, the envisioning and the QAs predict possibilities without necessarily involving the exact parameters of the possibility. The nature of the ambiguity may be eliminated or grossly changed by the global experts and the QAs so that the ambiguity is so simplified that its RALCMs no longer mention many of the possible parameters. These are the cases for which there is no qualitative ambiguity yet the quantitative parameters remain unknown. For example, global experts may have eliminated the possibility that the object reaches C2 because the slope is vertical there.



Hence the object must fall off on S1 and there is no ambiguity. In order to determine the exact point of leaving the surface the original ambiguity on S1 generated by the local experts must be used. The result is the collection of RALCMs used to resolve FLY-SLIDE. In some cases the local experts may not predict an ambiguity. One such case is SLIDE and that is handled as a special case (the SLIDE-SLIDE RALCMs are used).

Now we can see more clearly the connection between the quantitative RALCMs and the original ambiguity generated by the envisioning. Basically there are two issues that arise in the roller coaster mini-world: the object can change its direction of movement or the object can leave the surface. The first possibility concerns velocity; the second concerns normal forces. These issues

are present everywhere, but on any given section of the curve very simple qualitative reasoning can determine exactly which issues are relevant. This is what envisioning does. Envisioning recognizes that an object will not fly off a flat surface. The envisioning recognizes the issues that might cause ambiguity and constructs an information structure including only that quantitative information relevant to the current issues. QAs using more global knowledge than the envisioner then use the constructed information structure to resolve the issues the envisioning determined to be important. It was the envisioning that determined that the normal force between the object and the surface was not relevant to any ambiguity that might occur there.

The general method for when solving for a point within the segment is to hypothesize a point. The conditions which must be met at this point are determined (e.g. zero velocity for a velocity reversal ambiguity) and the position of the point solved for within some coordinate scheme.

The actual mechanics problems that have to be solved for each of the ambiguities are as follows:

SLIDE-SLIDE:

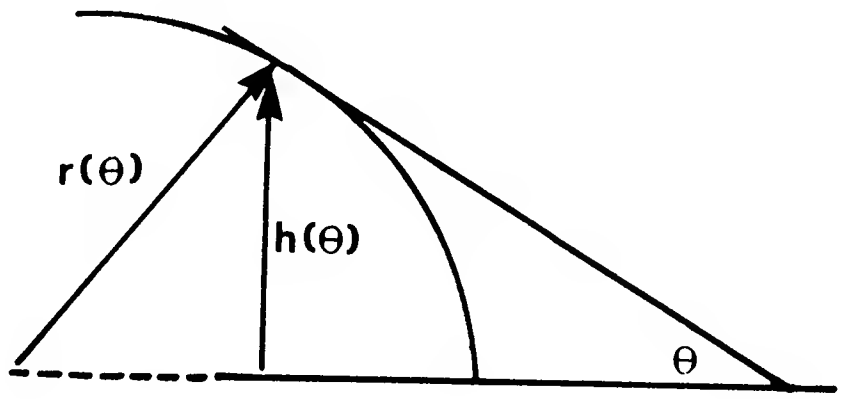
The final velocity at the endpoint of the surface should be calculated. If this velocity is positive no velocity reversal has taken place: if the velocity is negative or imaginary a velocity reversal has taken place. (This simple strategy will have to be modified and be much more sophisticated about the meaning of positive, negative and imaginary. That will only become apparent when we examine the mathematical expertise required to solve the equations and will be left for the next chapter.) If the exact point of velocity reversal is desired, then a hypothetical point is created at which the velocity is zero. The position is then solved for. Note that if a velocity reversal takes place the final velocity will be equal to initial velocity, but, in the opposite direction. The global principle embodied in this observation will be seen later.

FALL-SLIDE:

The curvature of the surface is keeping the object in contact as it moves along the bottom of

the surface. The object remains in contact with the surface as long as the force exerted by the constrained movement exceeds the force of gravity. The difference between these two forces is the normal force of the surface on the object. If this normal force becomes negative the object leaves the surface. Currently NEWTON assumes that this normal force changes sign when it reaches zero (except for the degenerate parabolas). NEWTON solves for the point at which the normal force becomes zero. If this point does not exist or would occur after the actual endpoint of the segment the object remains on the surface. Otherwise the object falls at the point at which the normal force becomes zero.

The following method can solve for that point in general. First the surface must be parameterized in terms of its angle θ, height $h(\theta)$ and radius of curvature $r(\theta)$. This parameterization is invertible and hence useful only because the envisioning has identified all the discontinuities and zeros so that θ must be monotonic over any segment.



The gravitational force:

$$f_g = m\, g\, \cos\theta$$

Force derived from constrained motion:

$$f_s = m\, v^2 / r$$

Normal force at surface:

$$f_N = f_s - f_g$$

The point at which the normal force goes to zero must be solved for:

$$m\ g\ cos\ \theta = m\ v^2 / r$$

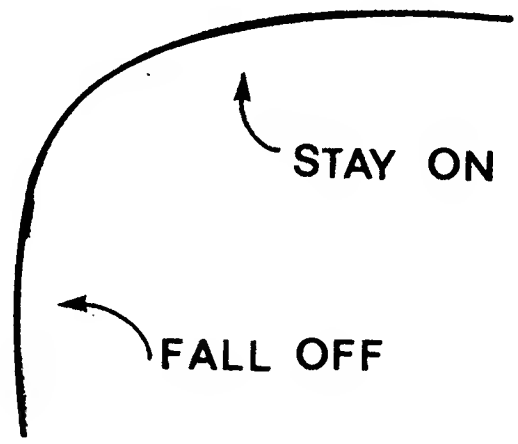
Conservation of energy can be used to derive the velocity:

$$v^2 = 2\ g\ h - v_{initial}^2$$

Which results in the final equation:

$$g\ cos\ \theta = (2\ g\ h(\theta) - v_{initial}^2) / r(\theta)$$

The actual parameterization of the surface must be calculated and substituted solving the above equation for θ or any other reasonable parameter. The applicability of this strategy depends crucially on how many zeros f_N has on the segment. If f_N has only one zero on the segment the strategy will succeed. This property depends both on the segment's length and on its curvature. The circle is, obviously, a curve for which the strategy succeeds. One example of a segment which does not meet the criteria is the following.



This ambiguity is an example where different strategies can be used depending on the desired goal. Because f_N is proportional to the square of the velocity, the property of remaining on the surface is independent of the direction of the velocity. Consider an object sliding down underneath a surface. If f_N increases monotonically only the top point of the surface needs to be checked: if f_N is sufficient at the top to prevent falling the object will not fall off anywhere on the surface; if f_N is insufficient the object will fall off somewhere on the surface. A sufficient condition for f_N to be monotonically increasing (allowing the applicability of the simple strategy) is that the radius of curvature decreases monotonically. That is not, however, a necessary condition (the vertical parabola is the limiting case). NEWTON knows this criteria for some curves. It can also be told explicitly in the problem. If f_N indicates the object will fall, the point of falling has not

necessarily been determined. Only if the top point was the initial point on the segment has the point of leaving the surface been determined. Usually, the more difficult strategy presented earlier must be used.

FLY-SLIDE:

The strategy for this kind of ambiguity is identical as that for the FALL-SLIDE ambiguity.

FLY-SLIDE-SLIDE:

This ambiguity can best be resolved in two steps. First a check is made ignoring the possibility of velocity reversal treating the ambiguity as if it were FLY-SLIDE. If the object does not fly off a further check is made to see if a velocity reversal occurs by treating the ambiguity as SLIDE-SLIDE. If a velocity reversal occurs it is not necessary to check to see whether the object falls off on the way back. The previous derivation for the FALL-SLIDE and FLY-SLIDE proved that remaining in contact with the surface depended only on the magnitude of the velocity, hence, if the object did not fall off on the way up it would not fall off on the way down.

4.4 The Data Base

The envisioner and the QAs examine features of the problem and make inferences. We have already seen how the envisionment tree is used to represent many of these inferences. Other inferences which are made are stored in an assertional data base. Although examples of its use will not be given until the next chapter it is the QAs that add most of the assertions to this data base. Some of the basic assertions that the QAs make will be given here.

The data base used in NEWTON is a derivation and simplification of the CONNIVER data base <McDermott & Sussman, 74>. Contexts are not used, and the data base is used only to store assertions. The two basic access functions are ASSERT and FETCH. ASSERT adds a datum to the data base. FETCH retrieves from the data base. The retrieval pattern may contain pattern variables in which case FETCH returns a list of a-lists. Two other auxiliary functions are used in

relation to FETCH. The function TRYNEXT is used to examine the a-lists returned by FETCH. The function TRYNEXT is called on the object returned by the FETCH and explicitly SETs the variables in the first a-list. The function FETCH1, which is the most common access function, first calls FETCH on its pattern and then if that FETCH only matched one datum calls TRYNEXT to instantiate the pattern variables.

The scene is described in the data base and some of the more important assertions are its description:

```
(HIGHER segment point1 point2)
```

```
(CONCAVITY segment type)
```

The segment joins two points. The first point is the higher of the two. To find the high point of segment S1 one would do the following fetch:

```
(FETCH '(HIGHER S1 !>highpoint !>NIL))
```

NIL as a pattern variable means that the value it matched should be thrown away.

As NEWTON solves a mechanics problem, it uses the data base to record its progress. One of the structures that gets built is the description of the disambiguated event. It should be noted that NEWTON stores physical properties of the event by their time of occurrence rather than their location of occurence. This is because the velocity at a point can be ambiguous. Assertions about sequential times are stored as:

```
(TIME-SEQUENCE time1 time2)
```

time1 comes directly before time2. Note that time on a segment is used to denote a generic time. When the QAs have not yet disambiguated all the possibilities at a point the following fetch returns multiple times.

```
(FETCH '(TIME-SEQUENCE time !>next-time))
```

Since the envisionment is a tree there is never any ambiguity about the previous time.

```
(FETCH '(TIME-SEQUENCE !>previous-time time))
```

This fetch returns only a single possibility. With each time the associated location is stored.

```
(AT object location time)
```

These are but a few of the more common assertions that NEWTON adds to the data base. Later, when quantitative knowledge is examined we will see how these assertions are explicitly used.

The internal design of NEWTON is such that most of the information that it builds is either attached directly to the envisionment or asserted in the data base. This makes it convenient to build more knowledge into NEWTON without having to be concerned with the details of all the other routines. It also explains why there are many more assertions in the data base than has been presented here.

4.5 Definite and Indefinite Questions

As was noted in the introduction of this chapter, mechanics problems generally consist of an event and a question about that event. The question itself can be divided between an aspect of the event and a request for a feature about that aspect. In the question "What is the velocity at point X?" the aspect of the event is when the object reaches X and the requested feature is its velocity. The analysis of the entire event using the strategy of the previous section can be used to answer the actual question. The resultant event structure is scanned for the presence of the requested aspect and then the necessary features the question requests are determined; however, if the question were examined earlier, before the event was completely analyzed, the work could often be considerably lessened. The question may only ask about a small part of the event so that not all the information needed to resolve the entire event needs to be present to resolve the partial event. If the aspect refers to a point that can be determined a *priori*, the question will be termed definite. "Will the object reach X?" is a definite question. "What is the velocity of the object when it falls off?" may be definite or indefinite depending on the places in the envisionment where the object might fall off. All indefinite questions must, however, become definite questions if they are ever to be answered. (Except in degenerate cases, where, the velocity at both possibilities can be determined to be equal.)

This section outlines a strategy to handle definite questions. Indefinite questions, which must eventually become definite, are handled as if they were purely descriptive questions until the

problem has been sufficiently reduced that they refer to definite points.

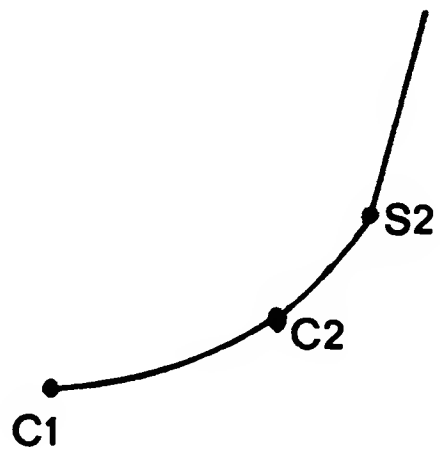
Whether the question is quantitative or qualitative, the possibility of reaching a definite point must be determined. A possible path must be discovered to the point. (Unfortunately, there may be many such paths.) This path and the identification of ambiguities within it constitutes a linear plan to solve the problem. A plan for the sliding block problem might be:

Solve SLIDE-SLIDE on S2 and if C3 is not reached fail.

Solve SLIDE-SLIDE on S3 and if X is not reached fail.

The same RALCMs that were used to answer "What happens?" type questions can be used to resolve these ambiguities in the plan. The results of the quantitative analysis will, of course, have to be treated slightly differently. Depending on the result of the analysis, analysis either proceeds on to the next statement as in "What happens?" questions or halts indicating the point was not reachable.

The plan may have branches in it.



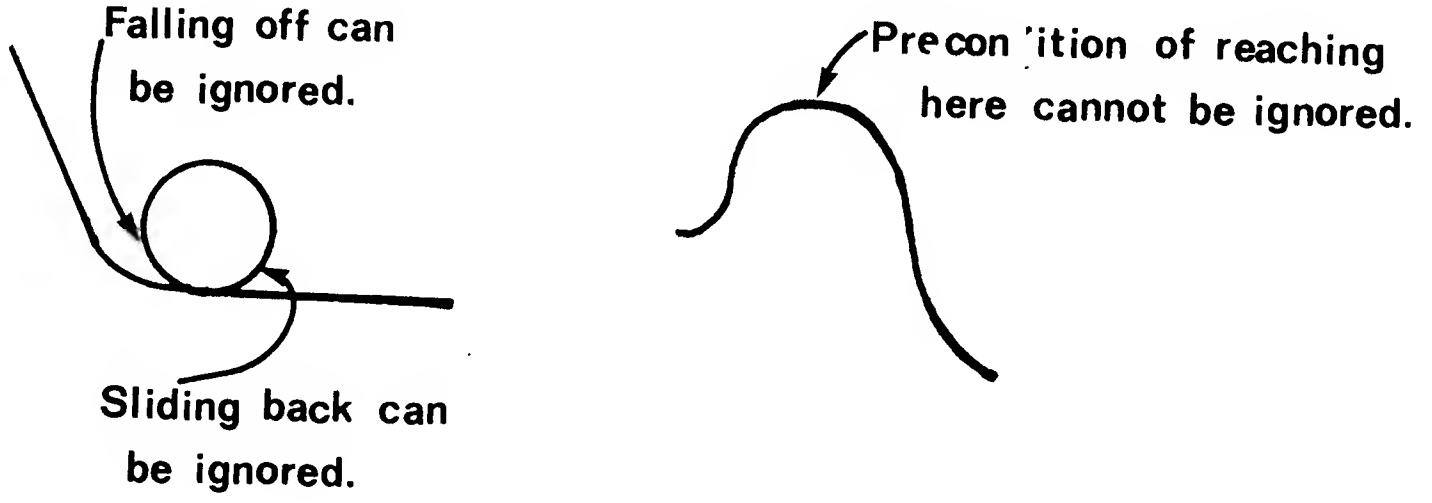
If the object were given an initial velocity to the right at C2, would it reach C1?

The plan returned would be different depending on whether a velocity reversal occurred on S2.

In the solution to the original sliding block problem, the first step in the plan, the SLIDE-

SLIDE step on S2, could have been ignored. In the loop-the-loop problem two steps in the plan were, in fact, never solved. Other information in the problem can be used to eliminate steps; that is not saying that the ambiguities do not need to be resolved. It says that with respect to the question asked certain ambiguities to not have to be explicitly solved. NEWTON contains a collection of strategies which examine the proposed plan and simplify it.

The most important of these strategies is recognizing which steps in the plan need not be examined. That is, failure to reach the goal would be detected whether or not that specific step in the plan were examined. In the sliding block problem a SLIDE-SLIDE step could be ignored and in the loop-the-loop problem a SLIDE-SLIDE and a FALL-SLIDE step could be ignored. This global strategy is useful mainly because of the existence of global quantitative knowledge, namely conservation of energy. The crucial question is when can steps be ignored and when are they important.



In the case of going over the peak to an ultimately lower position the global strategy of calculating final velocity gives a positive, yet false, solution. The applicability of such global strategies depends critically on local features of the area they span. Often a much simpler analysis, envisioning, can identify these relevant features. The height of the hill cannot be ignored because for any given initial velocity there exists a hill which the object would fail to climb. Another example of a local precondition for a global strategy is that a path exist in the first place. The global strategy of energy must take into account whether there is local friction or local non energy preserving events.

Let us examine the plan for the hill traversal problem to see what kinds of steps can be ignored and which cannot.

SLIDE-SLIDE

FLY-SLIDE-SLIDE

FLY-SLIDE

In this problem and the loop-the-loop problem a sequence of SLIDE-SLIDEs appears, the first of these can always be eliminated. Basically SLIDE-SLIDE only calculates the final velocity so that SLIDE-SLIDE followed by any other step requiring the calculation of velocities can be ignored. Note, however, that this analysis works only if no other expertise has eliminated some possibilities. If some possibilities have been eliminated by other expertise or by externally given advice this information must be taken into account so that plan steps are not incorrectly deleted. For example, the sequence SLIDE-SLIDE followed by FLY-SLIDE is an impossible sequence and should not be amalgamated, since the intermediate step of FLY-SLIDE-SLIDE is missing.

There are many other reasons why steps can be eliminated. The elimination of the FALL-SLIDE step in the loop-the-loop solution gives us two more elimination strategies. The first is that even though the envisionment divided the circle into four different segments, these segments were still in many senses one complete whole -- a circle. The crucial point for falling off can be identified to be the top, and this was how the problem was solved. The other possibility was the employment of symmetry. The first two steps of SLIDE-SLIDE and FALL-SLIDE could have been amalgamated into FALL-SLIDE and, using the special strategy for CIRCLEs presented in the previous section, the normal force would be calculated at the top to determine whether the object reached the top. Since the circle is symmetrical about the vertical and the object remained on the surface on the way up, it will remain on the surface on the way down.

4.6 Boundary Conditions

A problem can specify the boundary conditions and ask about the resultant effect (e.g. the sliding block problem) or it can specify a resultant effect and ask about the necessary boundary

conditions to produce it (e.g. the loop-the-loop problem). In all of the examples discussed so far the boundary conditions are the initial conditions. The previous two sections have discussed, at length, how from given initial conditions the event can be traced forward through time. Tracing back through time to find which conditions must of held true for the desired final result to hold can be handled using almost the same strategy. (We will assume we are dealing with definite questions.) To progress forward in time results were calculated and tests made on these results to determine how to proceed. In progressing backwards in time these tests are constraints that the initial conditions must satisfy. (If there were disjunctions in the forward plan, there will be disjunctions in the constraints on the initial conditions.) The loop-the-loop problem, although treated as an initial condition problem, was solved using this strategy.

There are other interesting ways to look at this strategy. The desired initial condition may be hypothesized using a variable, then calculation proceeds adding constraints to that variable until it is finally determined. The current mini-world is independent of the direction of time so the entire problem could be run backwards using the desired effect as the initial conditions. With such a strategy all problems can be solved by forward reasoning through time only. Unfortunately, this strategy is only convenient when the desired final result implies only a single initial constraint to build the hypothetical initial conditions. All of these possible methods are isomorphic in that they all lead to identical kinds of quantitative analysis no matter how different the qualitative analysis for them appear to be. NEWTON currently uses the strategy of hypothesizing parameters (variables assumed to be known but without values) for desired initial conditions and determining what constraints resultant conditions impose on them.

4.7 Limitations

This chapter has illustrated the basic question types and has laid the groundwork for a connection between the envisioning of the previous chapter and the quantitative knowledge of the next chapter. Chapter 6 will show many detailed examples of the interaction of these pieces of knowledge.

We have seen the kinds of questions NEWTON can answer with the strategies outlined in this chapter. The following are some examples of the kinds of questions which NEWTON cannot yet deal with.

"Which path is the fastest?"

"How can the path be made faster?"

"Construct the fastest path?"

"Where does the object go the fastest?"

"How many times does the object cross S1?"

"What is the acceleration of the object on a flat segment?"

"What is the period of oscillation on a cycloid?"

"Why is the acceleration on a flat surface constant?"

"Does the mass of the object affect the final velocity?"

NEWTON contains sufficient information to solve most of these problems, but it does not have enough QAs to understand these questions.

5.0 REPRESENTATION OF QUANTITATIVE KNOWLEDGE

5.1 Desiderata for a Quantitative Representation

The envisionment produces an initial plan for solving the problem. This plan is subsequently analyzed by the QAs and if the possibilities within the event cannot be disambiguated or if the problem explicitly asks for a quantitative value, quantitative knowledge is used. The representation of this quantitative knowledge is the topic of this chapter.

That quantitative knowledge is just another kind of knowledge available to NEWTON enforces many constraints and allows for many freedoms in its design. The discussions concerning the QAs have described the interaction between quantitative knowledge and the rest of NEWTON. The envisioning and planning that has proceeded any problem posed to the quantitative knowledge permits it to ignore problems of searching large spaces. The rest of NEWTON communicates with the quantitative knowledge through the top level RALCMs and the data base. The QAs reference the top level RALCMs and any part of NEWTON can access the information that the quantitative knowledge adds to the data base.

The dichotomy between qualitative and quantitative knowledge has become rather obscure as it was ascertained what kinds of knowledge could be considered as qualitative. Nevertheless it will be useful to re-examine this dichotomy as we begin to investigate quantitative reasoning. Qualitative reasoning involves studying features of the problem while quantitative reasoning is characterized by assigning variables to the features observed by qualitative reasoning, ignoring the physical meaning of the variables, artificially manipulating equations consisting of these variables, and reassigning physical meaning to the results. This loss of meaning of variables leads to abstraction in the manipulation process and gives mathematics its generality and power. Unfortunately, this loss of meaning makes the transference and selection of relevant equations very susceptible to error. Quantitative knowledge handles these transferences and the selections of equations.

The key to understanding the nature of quantitative knowledge is to examine how people solve problems. People rarely use general methods to solve problems. Instead, the knowledge is

divided into a heirarchy of special cases such that the "simplest" applicable strategy is used. This is in great contrast to a general strategy which uses the same strategy regardless of the problem. (A good example of a general strategy is the electronic circuit simulator which applies the same strategy to one resistor as it does to a complete amplifier.) The major portion of a mechanics text consists of a description of the heirarchy of special cases; hence, once the representation is developed, the pieces of knowledge can be obtained directly from the text.

In NEWTON the quantitative knowledge is organized into a collection of interrelated pieces called RALCMs. Each RALCM specializes in certain aspects of the general problem being fixed. Some examples of RALCMs might, for example, be the principle of conservation of energy or the trigonometric relationships within a triangle.

A major demand on this representation is the ability to to express the conditions for which the strategies are applicable. A considerable part of any RALCM will then contain information about when it is applicable (or which sections of it or other RALCMs are applicable). These RALCM descriptions are examined by an interpreter, IRALCM, to answer questions. A RALCM is invoked when IRALCM tries to apply it to solve a problem. RALCMs can only be invoked for the purpose of discovering the value of a particular variable. This purpose must be made explicit when the RALCM is invoked. The invoker of a RALCM requires the value for a certain variable and knows that the RALCM it invoked knows something about that variable.

When a RALCM is invoked the search for information about the desired variable will include every possible piece of the knowledge in the RALCM. Only after that search fails will it return with a detailed explanation about that failure. This is based on a theory, for which we see some evidence in people solving mechanics problems, of trying to use knowledge local to that aspect of the problem currently being examined. Of course, this depends crucially on the representation. Consider the example of a right triangle RALCM given the base and the altitude and asked for the hypotenuse. It might find an equation relating the sine of an angle and the altitude to the desired hypotenuse, but it should not immediately complain that it needs the angle and instead search further, discovering the pythagorean theorem. This seemingly unimportant decision to maintain the

localness of interaction within the representation has, as we shall see, momentous consequences for the internal structure of IRALCM.

5.2 Some Consequences of the Desiderata

The desiderata of the previous section lead to many consequences for the organization of IRALCM and the actual representation of the RALCMs. There are many ways the theoretical principles can be employed. The following is a description of how they are used in NEWTON.

When a RALCM is invoked, IRALCM looks at the purpose for the invocation and then examines the RALCM to see where in its description it makes a possible assignment to that variable. This possible assignment can either be an explicit assignment or a reference to another RALCM. This assignment may have a number of prerequisites which have to be met. Many possible assignments may be found, each with different prerequisites. One of these must be chosen. Much of the syntax for RALCMs involves how such preconditions can be structured.

RALCMs act at various levels of the problem solving process: one RALCM might be concerned about using an energy method versus a Newton's Law method while another RALCM might be concerned about the base length of a sliding block. In a usual LISP-like representation we would either have to pass down all relevant variables from the top-level or use free variables. Passing variables down from higher levels has the problem that the top-level RALCMs have to be concerned about many details which are irrelevant to them. Free variables have the problem that more than one RALCM must know their names. The RALCMs we propose here are not invoked with argument lists, neither do they explicitly refer to free variables. Instead, the name of a variable is only local to the RALCM that defines it. Communication is achieved by assigning meanings to every variable and constructing bindings between variables with matching meanings. These meanings, which are stored in the data base, can then be accessed by both other RALCMs and other parts of NEWTON. This solves the problem of variable names and makes it possible for the other knowledge to examine the state of a quantitative computation.

Any RALCM under consideration by IRALCM has an environment. This environment

contains the current variable bindings (A-LIST) and the current state of the computation. The same RALCM may be under consideration with a number of different environments. A RALCM combined with a particular environment is an instantiation of that RALCM. So far this paper has often used RALCM to mean an instantiated RALCM; the context of the word RALCM will indicate which meaning is intended.

A record is kept of all computation that is done within an instantiation. When a RALCM is unable to meet its purpose it fails and explains why it failed. The invoking RALCM can either try to resolve the complaint or, as is often the case, just ask the same RALCM a different question. Any computation that was done on the first purpose will not be repeated. If the new purpose was a complaint of the old purpose the RALCM can immediately return because it has already done everything possible to resolve that purpose. Note that having asked the instantiation for two purposes does change the fact that two different complaints have been returned and superior RALCMs will try to resolve either of these complaints.

Let us examine a RALCM. For the moment we will ignore the syntactic details of the description.

```
(DEFINE-RALCM  KIN (?OBJECT ?SURFACE ?T1 ?T2)
                   ((VF (VELOCITY ?OBJECT ?T2))
                    (VI (VELOCITY ?OBJECT ?T1))
                    (D (DISTANCE ?SURFACE))
                    (T (TIME ?T1 ?T2))
                    (A (ACCELERATION ?OBJECT)))

        (EQUATION E1-KIN
           (= VF (+ VI (* A T))))
        (EQUATION E2-KIN
           (= (* VF VF) (+ (* VI VI) (* 2 A D))))
        (EQUATION E3-KIN
           (= D (+ (* VI T) (* .5 A T T)))))
```

It is important to bear in mind that RALCMs are not procedures in any sense, instead they describe relations among variables. The '?' variables of the first list of the RALCM body are the primary variables which must be supplied when the RALCM is invoked. The primary variables describe the objects about which the RALCM is concerned. The next list is the secondary variable A-LIST. Each of its elements describes a secondary variable. The first element of the pair is the

atom which is used in the later equations. The second element of the pair is the global binding which must be expanded by substituting in for the primary variables. Whenever a RALCM is instantiated it must be given an instantiation pattern, a short description of the scene about which the RALCM is solving problems. For example the KIN RALCM is given a four-tuple indicating the object about which it is solving kinematic problems, the surface this object is moving on and the times of this movement. This pattern serves two purposes. First, it gives a name to every instantiation so that all references to a RALCM with the same scene will be to the same instantiation. So, the same instantiation can have multiple invokers. Second, this name is used to give meaning to the secondary variables. If a secondary variable is to have a nonlocal binding it must be given a meaning which can match with other meanings. These meaning descriptions contain primary variables which must be assigned the names of the specific objects under consideration. In the description of the secondary variable VF in KIN, the primary variable ?SURFACE must be assigned the value S1. The values for the variables are derived from matching the pattern at the beginning of the RALCM with the instantiation pattern.

Since the meanings of these secondary variables cannot be elaborated unless all the primary variables have assigned values, a RALCM cannot be instantiated unless all of its primary variables are given values. The formats for instantiation patterns and meanings are very simple. Patterns and meanings must match exactly or they do not match at all. The format of instantiation patterns is arbitrary, but the invoker and invokee both must know this format. The descriptions of variables have fixed formats and a variable can only be of certain types. A meaning description consists of a type such as VELOCITY or ACCELERATION and an ordered list of modifiers which indicate such information as the name of the moving object or the time span of the movement.

The data base is used to keep track of variable meanings and instantiation patterns. Whenever a new secondary variable reference is made the meaning description is elaborated by substituting for the primary variables and then looked up in the data base. So the A-LIST for an instantiation contains local variable name - global data base pointers, not local variable - value

pairs. It is the data base that really describes the current environment and not any A-LIST or stack structure. Thus, unbinding in the conventional LISP sense becomes a vacuous concept.

RALCMs can reference other RALCMs, indicating that the referencee knows something relevant about the topic of the referencer. Every time a reference link is created, the variable meanings for the referenced RALCM are matched with the data base. (In the remainder of this chapter variable will be used to refer to secondary variable unless explicitly indicated otherwise.) Added to the property list of a variable is the fact that there is a new RALCM which knows something about that variable. Associated with that fact is a reference path or plan for how to reach that instantiation. So whenever a value for a particular variable is required its property list can be examined to find where it is referenced. If all these references fail, other RALCMs can be looked for in other referencable instantiations and new references created.

Whenever an individual attempt to find a value for a variable fails there is the choice of trying to deal with the failure or trying another alternative. Earlier we have taken the general position that computation should remain local for as long as possible. This means that alternatives outside (above) an instantiation should be considered only after all attempts to resolve the failures have also failed.

This control structure will be clarified with an example problem for the KIN RALCM. Suppose KIN was invoked for the purpose of discovering A with VF and VI known. KIN might return a complaint list consisting of items:

E1-KIN: KNOWN(T)

E2-KIN: KNOWN(D)

E3-KIN: KNOWN(D)∧KNOWN(T)

If the reader desires a more detailed scenario of how these RALCMs interact to solve a mechanics problem the sliding block problem presented in chapter 2 should be re-examined.

5.3 Basic RALCM Syntax and Semantics

A detailed description of the syntax of RALCMs will be presented. It will be shown how the semantics meet the conditions of the previous discussion. Some example RALCMs about physics will be shown.

A RALCM definition consists of a list of specifications and a body:

(DEFINE-RALCM <name> <pattern> <variable list> <body>)

<pattern> is a list of primary variables, upon instantiation these will be bound to values which will be used in the expansion of variable meanings. The <variable list> consists of all the variables that can possibly be bound into the data base.

<variable list>: $((\alpha_1\ \pi_1) \ldots (\alpha_n\ \pi_n))$

The α_i are the names of these variables and the π_i their corresponding descriptions. A variable description consists of a list whose first element is a type and the remaining elements, primary variables. The types that will be dealt with in the examples of this chapter are:

(ACCELERATION object)

(VELOCITY object time)

(DISTANCE surface)

(TIME time1 time2)

(HEIGHT surface)

(BASE surface)

(ANGLE1 surface)

(ANGLE2 surface)

The <body> is formed from a sequence of forms. Abstractly these forms are an unordered set of nonconflicting pieces of knowledge about the variables. The forms can be considered in any order or in parallel. IRALCM decides which of the forms should be considered and in what order.

The most basic form relating variables is the EQUATION:

(EQUATION <name> <expression>)

<name> is a tag NEWTON will use to refer to this equation, if omitted, a name will be created of

the form E0001. <expression> is an arithmetic expression in conventional LISP notation (or using the well known abbreviations) expressing a mathematical relation between the variables. Unless the <expression> contains an EQUAL (or =) it is assumed that <expression>=0. Usually the equation has to be explicitly solved for the desired variable. If this form fails it returns a complaint list just as a conventional RALCM would. We have already seen a simple kinematics RALCM. As an another example consider a possible RALCM for a right triangle:

```
(DEFINE-RALCM  RTRI (?TRIANGLE)
                    ((H (HEIGHT ?TRIANGLE))
                     (L (BASE ?TRIANGLE))
                     (HYP (DISTANCE ?TRIANGLE))
                     (T1 (ANGLE1 ?TRIANGLE))
                     (T2 (ANGLE2 ?TRIANGLE)))

          (EQUATION E1-RTRI
              (= HYP (SQRT (+ (* H H) (* L L)))))
          (EQUATION E2-RTRI
              (= (SIN T1) (// H HYP)))
          (EQUATION E3-RTRI
              (= (SIN T2) (// L HYP))))
```

It is possible to explicitly give a variable a value:

```
(VSETQ <variable> <expression>)
```

The VSETQ will fail back with a complaint list if any of the variables <expression> references are not known. Knowledge about an object moving along a straight line might be given by:

```
(DEFINE-RALCM  STRAIGHT (?OBJECT ?SURFACE ?T1 ?T2)
                         ...
                         (VSETQ A (* $ G (SIN THETA)))
                         ... )
```

So if only the angle was given the acceleration could be deduced, but the angle could not be computed given the acceleration. In the above case one would normally have used an EQUATION but the example illustrates in what kind of role the VSETQ can be used. Shortcomings in the mathematical expertise, which will be discussed later, will require it. The VSETQ can be used to evade problems of redundancy. If the equation is unsolvable by the mathematical expertise, a VSETQ can be used either to explicitly give NEWTON the solution or prevent it from trying to solve something for which it does not have sufficient expertise.

To reference another RALCM the RALCM form is used:

(RALCM <name> <instantiation-pattern>)

<name> is the name of the referenced RALCM and <instantiation-pattern> a list of primary variables describing the scene which will become the name of that instantiation.

Local variables can be created by the FPROG form. The syntax of the FPROG is similar to that of a DEFINE-RALCM definition:

(FPROG <variable list> <body>)

In order to express dependence relations between assignments the FCOND form is used.

(FCOND (<condition> <body>)

...

(<condition> <body>))

Only the first <body> which has its corresponding <condition> met is considered.

A RALCM passes through two phases, an instantiation and a subsequent interpretation. During the instantiation process variable meanings are expanded. RALCMs can include forms which are examined only at instantiation time. These forms can only reference the primary variables. The purpose of these forms is to make simple deductions about the primary variables and deposit their results in tertiary variables. Tertiary variables are prefixed by '@' and are treated identically to primary variables except that they cannot appear in the instantiation pattern of the RALCM.

The <condition>s of an FCOND are presumed to be evaluated at instantiation time. An explicit instantiation time evaluation is effected by the EVAL statement:

(EVAL <variables> <body>)

The <body> consists of a list of statements to be EVALed by LISP. The result of the EVAL statement is the result that LISP returns. The <variables> is a list of primary and tertiary variables which <body> references. Any changes that are made to these variables by <body> during its evaluation by LISP becomes permanent. Although the RALCM is by no means a procedure, at instantiation time it is scanned as though it were, executing the EVAL statements and

<condition>s in the conventional LISP order. An EVAL is used within ENERGY to determine the sign of the height difference:

```
(DEFINE-RALCM  ENERGY (?OBJECT ?SURFACE ?T1 ?T2)
                      ((VI (VELOCITY ?OBJECT ?T1) POSITIVE)
                       (VF (VELOCITY ?OBJECT ?T2) POSITIVE)
                       (H (HEIGHT ?SURFACE) POSITIVE))
          (EVAL (?PT1 ?PT2) (FETCH1 '(AT ?OBJECT !>?PT1 ?T1))
                            (FETCH1 '(AT ?OBJECT !>?PT2 ?T2)))
          (FCOND ((FETCH '(HIGHER ?PT1 ?PT2))
                  (EQUATION (= (* VF VF) (+ (* VI VI) (*  2 $ G H)))))
                 ((FETCH '(HIGHER ?PT2 ?PT1))
                  (EQUATION (= (* VF VF) (+ (* VI VI) (* -2 $ G H)))))))
```

The body of an EVAL can include arbitrary LISP expressions, but most commonly they are simple requests of the data base. The basic primitives to access assertions from the data base have been discussed in chapter 4. A ' " ' prefixing a pattern indicates that any forms within it which are prefixed by a ' , ' must have their current values substituted before the fetch is done.

Some minor modifications to the syntax of RALCMs will be introduced later as the need for them becomes apparent.

5.4 An Implementation

The system as specified so far still has many unelaborated areas and many different implementations are possible. In this section an implementation will be presented. This implementation has many problems and some ideas will be presented later on how to improve it.

In order to use a RALCM it must first be instantiated. The instantiation process builds an instantiation record so that later requests for variables can be handled. To instantiate a RALCM, its primary and secondary variables are expanded and a collection of plans are constructed about how to achieve the values for the secondary variables. As these plans are being built up, all the necessary EVALs are done, including the <condition>s of FCONDs. These plans are built recursively by invoking other RALCMs which are referenced through RALCM statements and including their instantiation record in the final resulting plans. The resultant instantiation record for a RALCM may then include plans for variables that were not even referenced in the RALCM.

For example if the MASS-MOVEMENT RALCM were instantiated on a flat surface it would return the following plan for (DISTANCE ?SURFACE)

```
(DEFINE-RALCM  MASS-MOVEMENT  (?OBJECT ?SURFACE ?T1 ?T2)
                              ((A (ACCELERATION ?OBJECT)))

               (FCOND ((FETCH '(CONCAVITY ?SURFACE ZERO))
                       ;if the surface is flat, try simple kinematics
                       (FPROG ((THETA (ANGLE1 ?SURFACE)))
                              (RALCM RTRI (?SURFACE))
                              (RALCM KIN (?OBJECT ?SURFACE ?T1 ?T2))
                              (VSETQ A (*  G (SIN THETA))) )))
               (RALCM ENERGY (?OBJECT ?SURFACE ?T1 ?T2)))
               ;energy will work for arbitrary shapes

(DISTANCE ?SURFACE)
          -> FPROG -> KIN  -> E2-KIN
                           -> E3-KIN
                   -> RTRI -> E1-RTRI
                           -> E2-RTRI
                           -> E3-RTRI
```

This plan is read: in order to find (DISTANCE ?SURFACE) the FPROG must be entered, within the FPROG KIN or RTRI can be used, within KIN the equations E2-KIN and E3-KIN mention the desired variable and within RTRI E1-RTRI, E2-RTRI and E3-RTRI mention the distance. (Although it is possible to EVAL the FCONDs later and include them in the plan, no use for such a construct has been found.) Once a RALCM is instantiated and the plans generated, the RALCM itself is never scanned again. The resultant instantiation record is a 3-tuple: the primary variable A-LIST, the secondary variable A-LIST and plans for every variable the RALCM and its references might know something about.

Before delving into the details of how IRALCM uses these plans it should be noted that the search through the plans is neither breadth first nor depth first. A crude complexity measure is instituted to insure that the mathematical derivations obtained are not too inelegant.

IRALCM is invoked on an instantiated RALCM and a variable. The plan for that variable is accessed from the instantiation record and is scanned tracing through the FPROGs and RALCMs until the leaf nodes of equations are reached. Each such equation is examined to see what other variables need to be known before that equation can be used. If other variables need to

be known an AND complaint is constructed. After all the substatements of a statement have been examined an OR complaint is constructed referencing all the complaints of the substatements (if the substatements were all equations these would be all ANDs). If the plan for (DISTANCE ?SURFACE) were being examined, these complaints would be:

```
E2-KIN -> (AND VI A VF)

E3-KIN -> (AND VI A T)

KIN    -> (OR (AND VI A D)
              (AND VI A T))
```

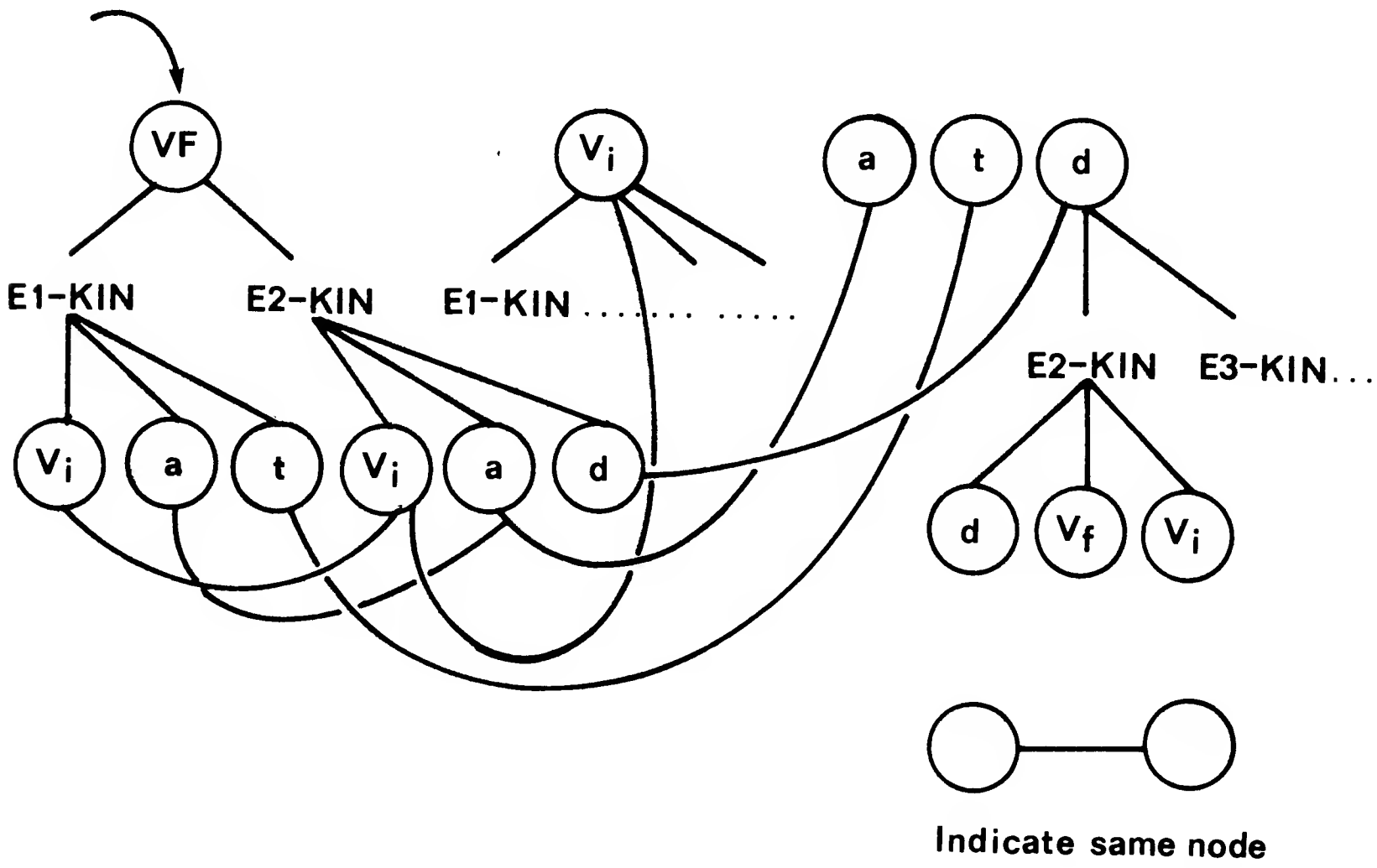
This procedure continues scanning the plan tree returning an OR complaint whenever there was an OR in the plan.

That would be the description of IRALCM if complaints were not handled. In accordance with earlier presented theory each complaint is scanned to see whether it can be resolved. In our example KIN might try to resolve its complaint by attempting to find a value for VI by using EI-KIN.

```
KIN -> (OR (AND (OR VI (AND  VF A T) A VF)
               (AND (OR VI (AND VF A T)) A T))
```

KIN would scan the entire complaint and make similar attempts to resolve A, T and VF. In the actual implementation the names of the statements generating the complaint would be included so that when values were discovered for the complained for variables it could also be used to indicate how computation should continue. This, of course, raises the issue of which variable request in the tree should be evaluated first when a value for a variable is discovered. If one were only interested in a solution, the order of processing the requests would not matter, but if a simple and elegant solution is desired, order is important. A crude complexity measure is used to order requests. The complexity of request is the number of equations which have to be examined before that request can ever be used in the final solution. The number of equations necessary to use a variable is approximated by the number of ANDs appearing above it in the tree. This crude complexity measure, although it fails to take into account the breadth of the tree, almost always succeeds in finding the simplest quantitative solution.

The same statement can be examined many different times for different variables. In fact, the same statement can be examined many times for the same variable. In order to make this reasonably efficient we would not like to scan a statement more than once for any variable. The problem with this strategy is that the complaint returned by a statement might depend on the examination of the problem so far: an expression can only be used once and it does not make sense to return a complaint containing a variable that is currently being searched for. One solution to this problem is to ignore the examination of the problem so far and only when the complaint is returned examine it for validity in light of the current examination of the problem. The same strategy that is used to prevent trying to achieve the same goal twice is used to prevent infinite recursion. Whenever a goal is started a check is made to determine if the goal has already been attempted or is in the process of being attempted. If the goal is started, a record is made of it. In both cases a partially built up complaint is returned. The complaint then becomes a graph and no longer a tree. The AND-OR graph for KIN with no knowns might be:



This AND-OR graph might seem a little large but it essentially represents all the mathematical relationships within KIN. It is independent of the examination of the problem so far and this graph could just as well be a complaint for any of VF, VI, A, T or D.

As the complaint is now a graph, the estimation of the complexity of a request within it becomes more difficult. Complexity is the depth of the minimum path from the top goal. Parts of the graph may later be resolved causing some links to disappear and elimninating minimum paths; thus, nodes are doubly linked so that whenever a node is deleted minimum paths that went through that node have to be recomputed.

Another problem that arises is fortuitous discovery. Suppose part of the graph has been generated and IRALCM, calculating elsewhere, discovers a value that the originally generated tree requires. That discovery could result in a sequence of discoveries within the partially generated graph possibly eliminating the point at which IRALCM is currently searching. The problem with the obvious solution of just immediately popping out of the section being searched is that the partial solution existing at the time is probably inconsistent. The partially existing graph could be used later if it could at least be made consistent. There are two options: (1) pop out destroying all the partial results determined so far or (2) complete the search until IRALCM naturally pops out. In the current implementation the latter approach is used because it is both easier to implement and desirable to prevent recomputation of partial results (although the final result will be thrown away).

The possibility of using the same equation twice or of solving a variable in terms of itself can be disregarded. Once an expression is used to obtain a value for a variable there need to be no concern about using that expression again because using the expression once indicates that all the variables it references are now known. Similarily an equation can not be used until all but one of the variables it references are known; the case of solving for a variable in terms of itself never arises.

On the surface the graph strategy may appear different than the theoretical outline we initially presented for RALCMs. If we examine closely we see that it does satisfy the theoretical principles. RALCMs attempt to solve the problem as much as they can within themselves and only

when they completely fail is a composite complaint returned. The complaints are clear statements of where the RALCM failed and includes information about how to proceed once its complaints are satisfied. The same instantiation can be used to request values for many variables, even though it might have succeeded in finding values for some of its variables. Furthermore, IRALCM keeps track of all computations made so that the same computation is never repeated.

This strategy enables NEWTON to carry through the quantitative analysis for the sliding block and the loop-the-loop problems of chapter 2. This implementation does have many problems and some modifications will have to made to handle these. These problems and their solutions are the topic of the subsequent sections.

5.5 Simultaneity and Redundancy

The implementation and the theoretical motivations for RALCMs presented earlier made one subtle oversight. In both discussions it was implicitly assumed that every quantitative problem could be solved by consequent reasoning alone. One way of looking at a VSETQ is as a consequent theorem: if the variable of the VSETQ is desired the variables in the equation can be checked and if they are not all known a complaint is generated which guides further search. An EQUATION is really just a set of VSETQs for each of the variables it references. The EQUATIONs and VSETQs can then be considered as consequent theorems, the order in which these consequent theorems are examined is carefully controlled by IRALCM. Nevertheless NEWTON encounters a difficulty that ordinary consequent theorem provers also run into. The mathematics of the problem establishes a set of constraints on the variables:

$$f_1(x_{11}, \ldots x_{1m'}) = 0$$

.

.

.

$$f_n(x_{n1}, \ldots x_{1m'}) = 0$$

Where m' is different for each constraint. IRALCM is given a goal x, it then looks at all the f_i

which mention x. If for some f_i all the other x_i except x are known, f_i can be used to derive x. Otherwise for every f_i which references x but also has unknowns x_i, IRALCM is applied recursively to obtain values for these x_i. Some sets of equations cannot be solved with this strategy:

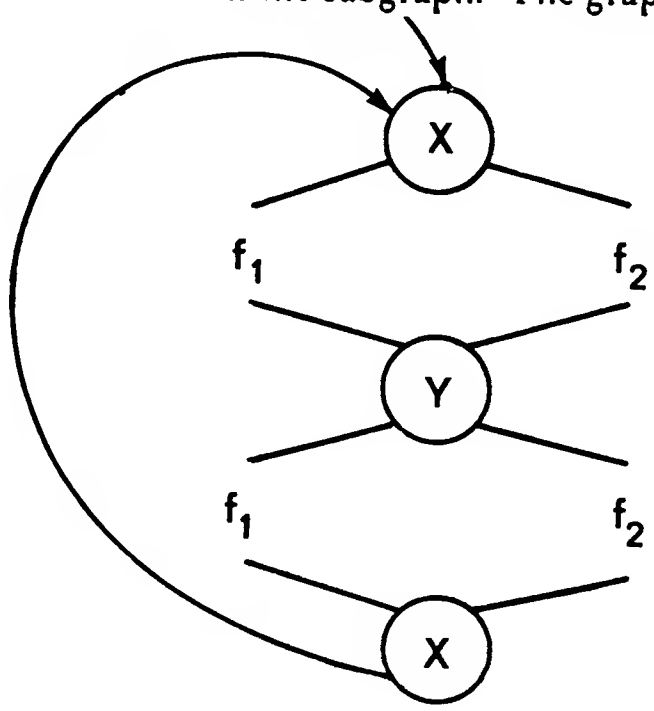
$$f_1(x,y) : x+y=0$$

$$f_2(x,y) : x-y=0$$

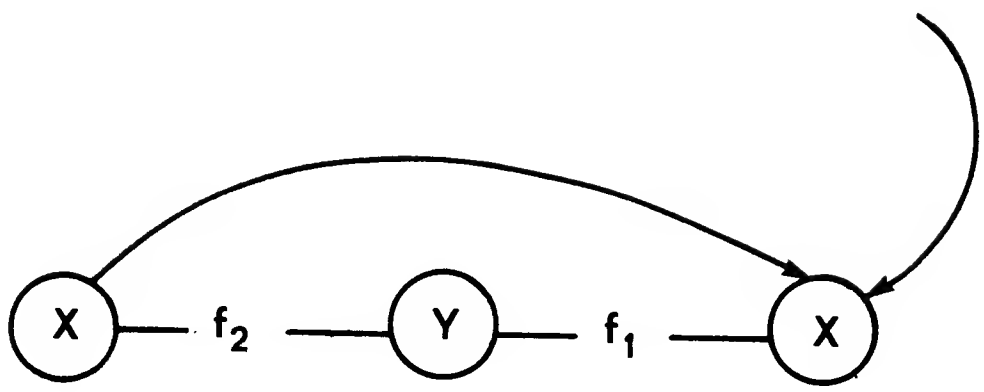
Searching for x, f_1 and f_2 are candidates, f_1 and f_2 both require y which in turn requires x so no solution is possible.

Antecedent reasoning, such as EL <Sussman & Stallman, 75> uses, is not of any help here either. In antecedent reasoning the goal is ignored and the constraints are scanned for any constraint that has only one unknown; then that variable is solved for. This process is repeated until no new discoveries can be made. If the desired variable is discovered the strategy succeeds. Since neither f_1 nor f_2 meet the criteria antecedent reasoning fails. This problem of simultaneity is a common problem with consequent or antecedent reasoning. EL solves the problem by arbitrarily hypothesizing a variable, NEWTON will also hypothesize a variable but the determination of whether simultaneity is present and which variable to pick will be decided more carefully.

A graph is solvable by simultaneity if it contains a subgraph which (1) includes every subnode of any AND, (2) contains one subnode of any OR, (3) includes the goal node, (4) uses every equation only once and (5) every node is either explicitly known or has a derivation in the subgraph. The last condition is perhaps the most crucial; essentially it insists that every variable must be derivable from the others in the subgraph. The graph for f_1 and f_2:



One subgraph meeting the criteria is:



Once such a subgraph has been constructed it can be used to solve for the desired variable.

The subgraph criteria of the previous paragraph is a necessary but not sufficient for a solution to exist. The most serious problem is that the criteria fail to detect redundancy. The following derivation in MASS-MOVEMENT traces out an apparently solvable subgraph. The desired goal is v_f and v_i=0 and θ is known (note that MASS-MOVEMENT applies only to a straight segment).

E2-KIN: $v_f^2 = 2\,a\,d$ needs a and d

E1-MASS-MOVEMENT: $a = g\,sin\,\theta$ known

E2-RTRI: $d = h / sin\,\theta$ needs h

E1-KIN: $h = (v_f^2 - v_i^2) / 2\,g$

SIMULTANEITY: v_f has a solvable graph

Solving the simultaneous system we get $v_f^2 = v_f^2$. This problem which satisfied our criteria fails to deliver any result. The difficulty is that the set of equations we are attempting to solve is redundant. Within the MASS-MOVEMENT RALCM there are many other examples of redundancies: the equations of RTRI and of KIN are redundant. Some collections of equations are redundant in all circumstances and others are redundant only when certain conditions are met. We have just seen an example of the latter. When equations are redundant only in certain environments it is probably because one is in fact a special case of the other. E2-KIN is actually a

special case of EI-ENERGY when MASS-MOVEMENT is applied to acceleration down an incline.

In order to indicate simultaneity a new RALCM language statement has been added:

(INDEPENDENT <integer> <eqn1> ... <eqnn>)

The <eqi> can either be normal EQUATION statements or they can be names of other equations. These names can also reference equations in other routines. If the name is an actual RALCM name all the equations of that RALCM are included in the INDEPENDENT statement one at a time (i.e. that is not to say that they are necessarily redundant with each other). The <integer> indicates the maximal number of equations that can be selected from within that set. On a straight segment MASS-MOVEMENT should instantiate the statement (INDEPENDENT 1 KIN ENERGY) indicating that one but not both of KIN or ENERGY should be used.

In the current implementation the redundancies are not actually examined when searching for a solution. Redundancies only cause problems when looking for possible simultaneities so knowledge about redundancies just adds another constraint to the algorithm that searches the graph for possible simultaneities.

The problem of simultaneity is not a consequence of our particular representation; it is an important property of mathematics and physics. Most of the basic laws of physics are simultaneous so to deal with simultaneity extra redundant formulations of the laws are often introduced. The resulting problem of recognizing redundancy is a difficult one. (Many a student has solved a complicated set of equations and obtained the well known solution "$0 = 0$.") After gaining some familiarity with a domain the student learns to recognize which equations are redundant. The recognition of which equations introduce redundancies in a problem involves knowledge about the causal physical reasoning behind them. Unfortunately, since this kind of reasoning is poorly understood, NEWTON is restricted to somewhat mechanical tests for simultaneity and redundancy.

5.6 Parameters and Elimination of Parameters

Strategies, similar to those used for recognizing simultaneity, can be used to deal with parameters and more importantly recognizing their possible elimination. Sets of equations may, on

the surface, appear to be unsolvable because some particular parameter is unknown, while the unknown parameters drop out when the equations are actually solved. In other cases parameters cannot be eliminated and a minimum number of parameters needs to be selected. Both of these problems can be dealt with in the simultaneity recognizing procedure by relaxing some of the constraints for acceptable subgraphs.

In the loop-the-loop problem the equation relating centripetal force and gravitational force had to be solved for v_f:

E1: $f_c = m\, v_f^2 / r$

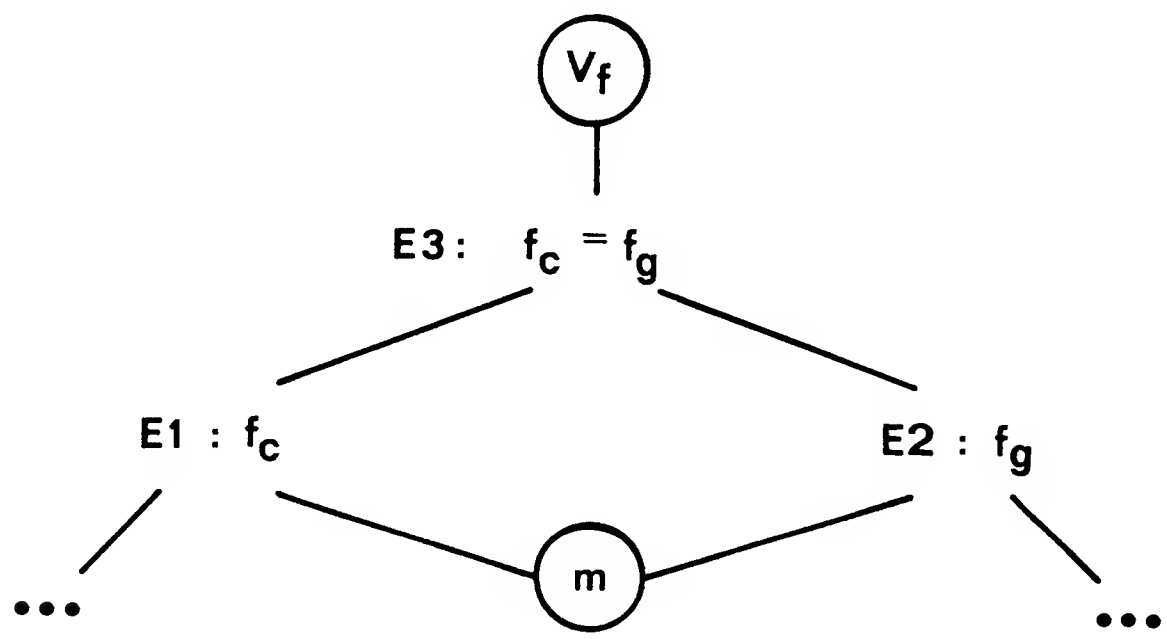
E2: $f_g = m\, g$

E3: E1 = E2

Superficially it appears that the mass m needs to be known, however, the equations are solvable for v_f.

$$v_f = (g\, r)^{1/2}$$

To recognize this possibility in a graph is quite simple. First, subgraphs containing leaf nodes should be allowed. From all such subgraphs, every path from an unknown parameter should eventually join with another (different) path from that same parameter. Each such join indicates that the variable could possibly be eliminated at that join. In our example the partial graph would look like:



This is a very trivial example but similar situations arise time and again in physics problems.

One more interesting example appears in <Polya, 62> in the section where he deals with puzzling counterexamples to the heuristic of first verifying whether the problem is reasonable or not:

A man walked five hours, first along a level road, then up a hill, then he turned round and walked back to his starting point along the same route. He walks 4 miles per hour on the level, 3 miles uphill, and 6 downhill. Find the distance walked.

Polya points out that information about the nonlevel part seems to be lacking and thus appears to be, on the surface, an unreasonable problem.

NEWTON would set up the following equations:

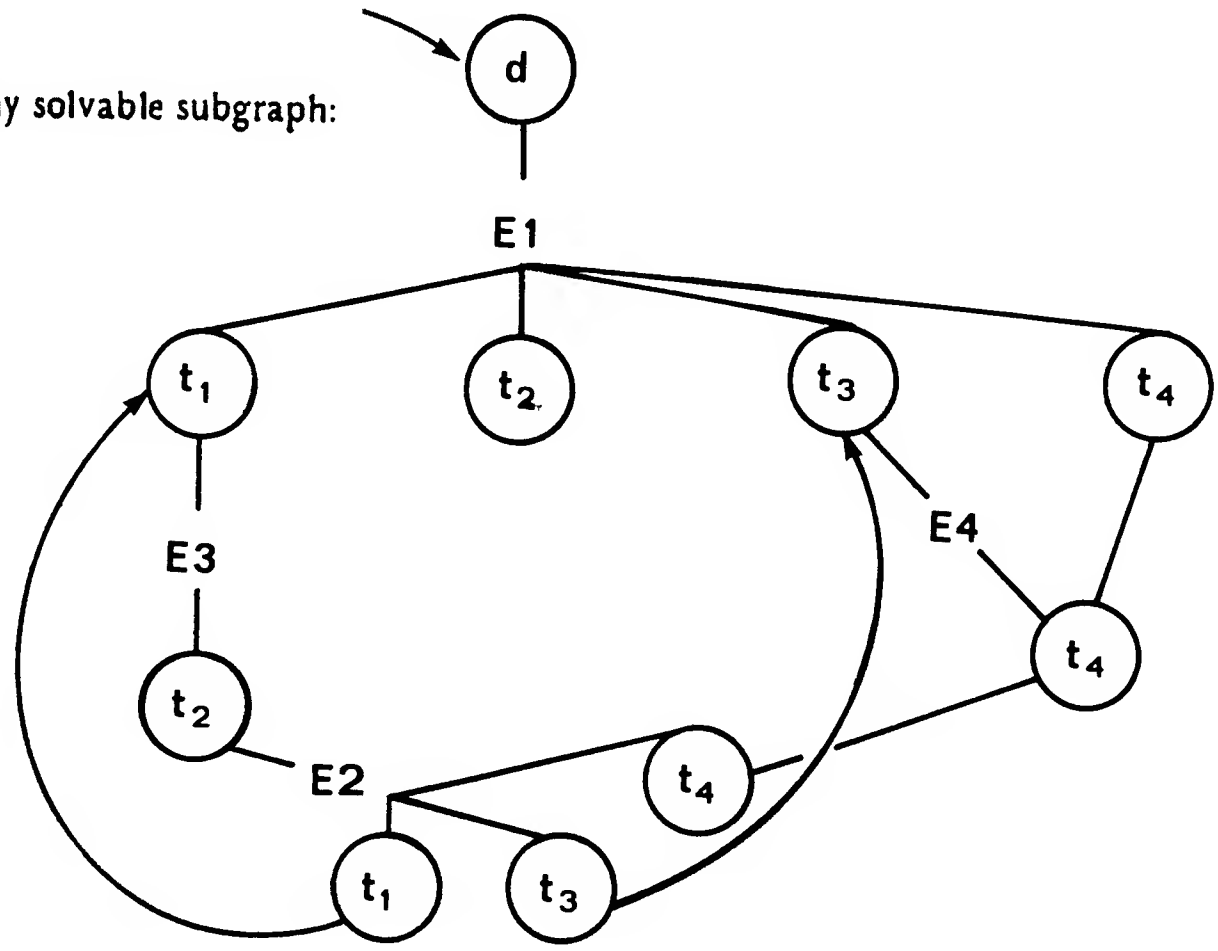
E1: $d = 4\,t_1 + 4\,t_2 + 3\,t_3 + 6\,t_4$

E2: $5 = t_1 + t_2 + t_3 + t_4$

E3: $t_1 = t_2$

E4: $3\,t_3 = 6\,t_4$

The equations have a possibly solvable subgraph:



t_4 although unknown may possibly be eliminated by E1. When the actual substitutions are done this is found to be the case.

With simultaneity the existence of the valid subgraph was only a necessary condition for a solution to exist. This is even more the case with this strategy for elimination of parameters which can only be eliminated by chance structure in the joining equation. The internal structure of the equation is completely ignored.

The strategy to discover minimal sets of parameters when elimination attempts fail should now be obvious. Just choose the subgraph with the minimum number of unknown parameters.

5.7 More Problems of Consequent Reasoning

The solution of quantitative problems seems to require purposeful directed search and thus is amenable to consequent strategies. Consequent reasoning eventually follows every possible path back from the goal. This often results in a proliferation of subgoals. The problem of proliferation of computation is also present in antecedent reasoning. This proliferation is rarely a problem in NEWTON since earlier planning has sufficiently reduced the search space.

From a global question the planning produced a plan of local steps each of which was easily solvable. Other subproblems such as those concerning the geometry of the scene do not have the benefit of similar planning. Without planning the entire problem is left to IRALCM. If the subproblem involves a limited search space it does not matter whether planning is done or not. It should not be surprising then that IRALCM may fail (compute excessively) for such problems.

The most obvious place this problem arises in the roller coaster world is in the computation of heights. If the problem involves only one segment the computation is locally to restricted to the RALCM (i.e. RTRI) for that segment type. With computing heights between arbitrary points an explosion of information can easily occur. The relations between heights of the points of the scene are:

$$h(x,y) = h(x,z) + h(z,y)$$

Where z ranges over all the known points.

That relation results in an incredible number of equations. If the scene contained n points ${}_nC_{2n}$ such equations result. Of these equations only C_{2n} are independent. One way to reduce the

number of equations is to introduce a hypothetical point G to which all heights are measured. The C_{2n} equations are:

$$h(x,y) = h(x,G) - h(y,G)$$

Note that this results in $n - 1$ equations for each $h(x,G)$. The elimination of parameters procedures can then be used to eliminate the heights to G. Suppose $h(C1,C2) = A$ and $h(C2,C3) = B$ and $h(C1,C3)$ is desired. The following equations would be involved:

$$A = h(C1,G) - h(C2,G)$$

$$B = h(C2,G) - h(C3,G)$$

$$h(C1,C3) = h(C1,G) - h(C3,G)$$

All the heights relative to G can be eliminated leaving the result:

$$h(C1,C3) = A + B.$$

The heights of segments may not be known and need to be computed, which heights should be attempted? The organization of IRALCM and the complexity measure it uses are inadequate. It would attempt far too many irrelevant subgoals and find inelegant solutions. Planning must be done before giving the problem to IRALCM. The solution to the general problem is difficult, fortunately this particular version of the problem has one simplification. The only experts (RALCMs) that know about heights involve the two endpoints of one segment. Those are the only subgoals worth generating.

In planning, the parameter elimination strategy is used before IRALCM invokes any of the local segment experts. The parameter elimination can be used at any time in the problem solving process except that in this case the exact points where that strategy fails to eliminate a variable can be used to guide further analysis. In the usual use these failures are thrown away. The procedure for finding a height is as follows: (1) the equations relating all the heights to G are set up, (2) the strategy for elimination of parameters is applied and (3) minimal sets of necessary parameters that could not be eliminated are set up as problems for IRALCM.

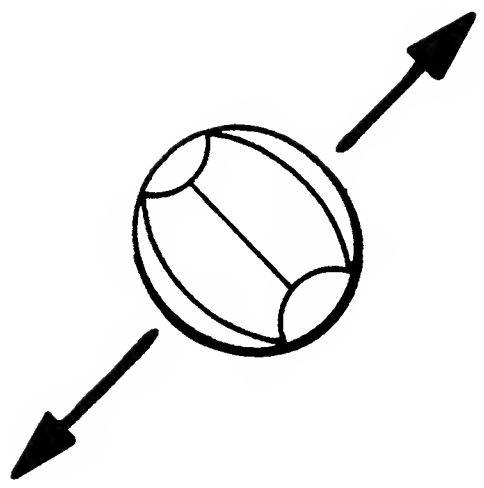
Consider another more natural strategy. The height between x and y is desired. Start with x. Find all points points z for which $h(x,z)$ is known and then try to find $h(z,y)$. Otherwise find the

two points adjacent to x and use the local experts to try to find the heights to these points. Instead of immediately invoking the local experts the rest of the path should be computed, and then that path requiring the least number of new heights should be attempted. This procedure is almost isomorphic to the previous one. The parameter elimination procedure does the equivalent work of searching for the z's and then trying the local experts. The only difference is that the former strategy can immediately deal with any other constraints immediately while the latter strategy must generate its plan and then let IRALCM include the other constraints. (An example of another constraint occurs in the case where the problem specifies that the sum of certain heights is a constant.) NEWTON currently uses the parameter elimination strategy, not really for any good reason, only that it is easier to implement as the tools are already all there.

5.8 Multiple Roots and Contradictions

The mechanics problem is eventually reduced to specific equations to be solved. Not all the information about the problem can be explicitly encoded in the equations, hence the mathematical solving of the equations searches a larger less constrained space for solutions. The mathematical equation solving then can generate solutions consistent with the equations it was given but inconsistent with other constraints in the problem. Equation solving produces such multiple solutions when it generates multiple roots. In order to select the correct root the constraints of the original problem have to be examined. This section deals with representing these constraints and then shows how these same constraints can be put to other uses.

As an example consider the simple projectile problem:



A person throws a ball into the air vertically (up or down). The ball is released at an initial velocity v_i and height h. When does the ball hit the ground?

The solution to this problem employs the kinematic relation:

$$d = v_i t + 1/2\ a\ t^2$$

In this case $d = -h$ and $a = -g$:

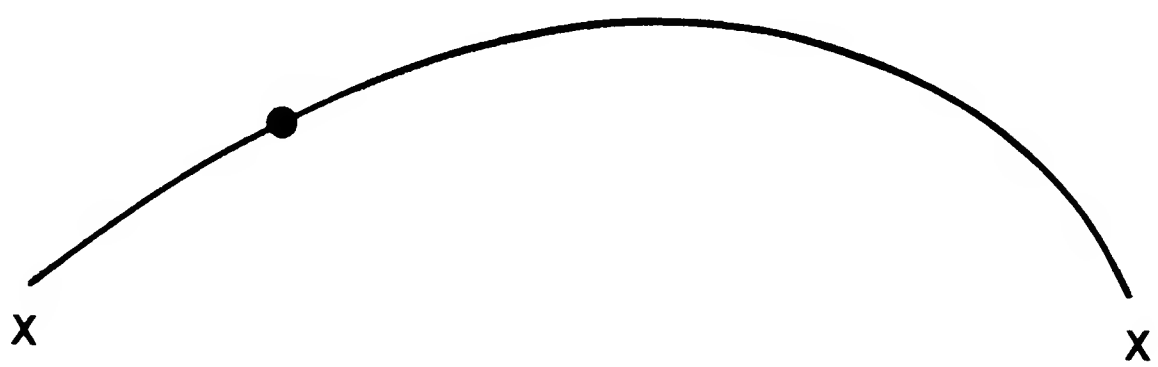
$$t = (v_i \pm (v_i^2 + 2\ g\ h)^{1/2}) / g$$

Ignoring for the moment the physics of the situation we find that the simplification of this expression is extremely difficult. It cannot even be determined whether the expression is a real number. If we were told that v_i was real, $g > 0$ and $h > 0$ then t could be determined to be real. There then are two possible real solutions:

$$t_1 = (v_i + (v_i^2 + 2\ g\ h)^{1/2}) / g$$

$$t_2 = (v_i - (v_i^2 + 2\ g\ h)^{1/2}) / g$$

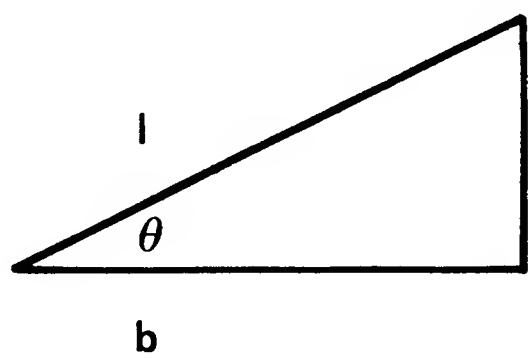
These correspond to the two physical solutions:



From the problem we know that $t > 0$ so $t = t_1$.

Any root selection problem involving a multiple number of roots can be handled by this method. If, however, the inverse function has an infinite number of values a slightly different strategy must be used. In a triangle we know that:

$$\theta = \arccos(b / l)$$



Without looking at the diagram the correct sheet of *arccos* cannot be chosen. From the diagram, θ must be acute, hence the correct sheet is determined. NEWTON tags the *arccos* according to which sheet it lies.

NEWTON currently relies on a very simple and primitive strategy to handle these difficulties. The strategy, although sufficient for most problems, would have to be substituted with a general inequality handler. Each variable and expression within NEWTON can be tagged with a range. Ranges can consist of a pair indicating what range the expression may assume:

$$v : (x . y)$$

If $x < y$ then $x < v < y$, if $y < x$ then $v < y$ and $v > x$ and if $x = y$ then $v = x = y$. The high end of the range may be *inf* and the low range *minf* which have the obvious meaning within the above inequalities. A range can also be *imag* indicating that it is a complex number. Otherwise x and y must be numbers. NEWTON contains procedures for performing arithmetic on ranges so that the range of an expression can be deduced from the ranges of its subexpressions. Not all possible ranges that can occur in an expression are expressible in the formalism. If the arithmetic is indeterminate the range of the result is *(minf . inf)*.

When a root must be selected the ranges of the desired result and each of the possible roots are determined. The range of each root is intersected with the range of the desired result. Every

root that results in a empty intersection can then be eliminated as a possibility. Assuming that v_i is negative in the first example NEWTON would have to choose (assuming that t was not zero):

$t : (0 . inf)$

$t_1 : (minf . inf)$

$t_2 : (minf . 0)$

NEWTON would select $t = t_1$. Already we see a weakness in our scheme, t_1 should have been determined to be positive, this deduction relies on the fact that v_i appeared in two places.

A similar process takes place in the case of other multivalued functions:

$\theta : (0 . \pi / 2)$

$arccos : (minf . inf)$

A specialist procedure would examine the range for θ and tag the expression indicating the sheet that should be used. The tag for *arccos* in the triangle case is *0*:

$$arccos_j x = arccos_0 x + j\pi$$

Where:

$$-\pi / 2 < arccos_0 x < \pi / 2$$

Every function that NEWTON knows about which has an infinite number of roots must have an expert associated with it which can determine from the imposed range what the appropriate sheet should be. These currently are the trigonometric functions: *arcsin, arccos, arctan, arccsc, arcsec.*

NEWTON can be told explicitly when presented with the problem what the ranges of the variables are. Even if a solution to a desired variable cannot be found, the ranges of the desired variables may be determinable and that might be sufficient to disambiguate among the possibilities. In the secondary variable list of RALCM definitions the range assumptions that are made are explicitly stated. For example, RTRI asserts that all the angles of the triangle are acute.

If a new range conflicts with an old range or if no root satisfies the range of the target variable a contradiction has occurred. This is the only place where a contradiction might be discovered. Contradictions are artifacts of making faulty assumptions. One common source of such contradictions is the elimination of plan steps. The elimination of plan steps often implies an

assumption that the outcome is as desired. Remember that plan steps could only be eliminated if later plan steps would detect their possible failure (i.e. probably detected as a contradiction). Assumptions can also come from other sources. One simple example is MASS-MOVEMENT's use of ENERGY. The implicit assumption is made that velocity reversal does not take place. The RALCM KIN, on the other hand, can be organized to give contradictions or not. Without ranges KIN would sometimes contradict and sometimes return a negative final velocity depending on what the givens were. If the final velocity in KIN were given a positive range MASS-MOVEMENT would always contradict if velocity reversal took place.

Question answering expertise uses such contradictions to disambiguate among the possibilities. Often a contradiction is the desired outcome. If the desired outcome is at variance with the actual outcome a global contradiction occurs. As we saw in chapter 4 such a contradiction forces processing on an indefinite problem back to the previous choice point and a global contradiction in solving a definite problem implies the originally posed problem was absurd.

5.9 Mathematical Expertise

The point at which physics knowledge stops and mathematical expertise begins is unclear. Much of the material of the previous sections bordered on this rather nebulous boundary. Nevertheless, it is important where that point is put in NEWTON. Furthermore, it is important to note why that particular point should be chosen, for after all, could we not have just given the collection of all relevant equations to MACSYMA and just let it solve the problem. In this section we shall endeavor to show that such a solution is not possible. One fundamental reason is just the recognition of relevant equations is very difficult. Another reason is that MACSYMA as it now exists is incapable of doing what we would want of it. MACSYMA is a collection of operators on mathematical expressions designed to be used interactively to solve problems, it is not a stand alone problem solver. The final reason, which determines the precise point of the boundary between mathematical expertise and knowledge about physics, concerns the design of MACSYMA itself: it is too large and some of the hooks and routines that we would desire are not present. In this section

we will elaborate these reasons, but first we will describe exactly what kind of mathematical expertise the current implementation relies on.

The most basic piece of mathematical expertise is simplification. When values and equations are substituted into other equations the resultant equation needs to be simplified to eliminate unnecessary variables and expressions. The crucial properties that NEWTON relies on in the simplifier are:

(1) That irrelevant variables be eliminated. (e.g. $e = x + 1 - x$ is independent of x so it should not appear in the simplified expression.)

(2) That if the expression contains only constants that the simplification process result in a single constant.

(3) The structure of the expression be modified to a form convenient for pattern matching, particularly for SOLVE.

NEWTON currently uses the general simplifier SIMP from MACSYMA and does not use RATSIMP or RADCAN. This simplifier cannot of course meet even the first criteria in the general case, however, it is sufficiently powerful for most of our mechanics problems.

The other piece of mathematical expertise that NEWTON relies on is the ability to solve a single equation for a variable. SOLVE depends critically on the simplifier SIMP. The same SOLVE is used for the mathematical manipulation required in simultaneity. SOLVE employs the MACSYMA pattern matcher SCHATCHEN <Moses, 67> to match for the possible patterns that can arise. SCHATCHEN is used to match for quadratics and linears in common expressions. SOLVE also knows about all the trigonometric functions and exponentials. SOLVE employs the strategies of the previous section to select appropriate roots.

Currently that is the sum total of NEWTON's mathematical expertise. If the expertise fails NEWTON can often reformulate the problem and resubmit it to SOLVE, but more commonly it just gives up. The reasons why more mathematical expertise was not used in NEWTON will now be outlined.

The single most important contribution of the RALCM organization is their encoding of the

appropriate dependencies and equations. Of the necessity of encoding such knowledge there is no doubt. The RALCMs also perform the crucial task of assigning the correct names to the variables so that no undesired conflicts and all desired coincidences occur. The first point at which there can then be an argument about where mathematical expertise could be used is after all the relevant equations have been recognized and instantiated.

At that point MACSYMA's ALGSYS could have been called to solve the problem. There are many objections to this. On the theoretical side it violates some of the basic design tenets of the RALCMs. There is no longer any sense of the local use of knowledge or of the originally intended control structure. The issues of complexity are lost as we no longer even know which equations were used in the derivation of the solution. On the practical side giving such a vast collection of equations would be extremely inefficient. ALGSYS is unable to deal with redundancy in the way NEWTON wants, nor is it able to generate any kind of complaints. ALGSYS does not contain the appropriate ability to select the correct root or attempt to eliminate parameters. It does, however, deal with the problem of simultaneity.

If we then agree not to use ALGSYS on the total problem and consider it only on single variable subproblems many of the same problems reappear. Also the routines MACSYMA uses to solve for even a single variable are so large as to be unwieldly. The problem of simultaneity is still with us and ALGSYS cannot be used for that either. Not any collection of equations is simultaneous and hence using ALGSYS to detect simultaneity would be extremely inefficient. If simultaneity is detected it would be inappropriate since the strategy, as a consequence of detecting the simultaneity, returns a graph which describes how the system of equations can be solved in terms of single step SOLVEs.

This leaves us with the necessity for a single variable SOLVE which relies on the MACSYMA simplifier and pattern matcher. Although the general simplifier is the weakest of the MACSYMA simplifiers, it is relatively small and self-contained. It will sometimes fail, but even the most powerful MACSYMA simplifiers cannot simplify some the expressions NEWTON generates.

This discussion should not be taken as an indictment of MACSYMA. As was noted in the

beginning of this section MACSYMA was designed to provide operators on mathematical expressions. It assumes that these operators can be treated as black boxes and are not designed with the extra hooks that NEWTON needs. MACSYMA does not know about either mathematics or physics. NEWTON knows about mechanics and wants to solve specific mathematical problems and to solve these problems it needs some knowledge about mathematics. Unfortunately, to be able to use the more powerful routines of MACSYMA it needs knowledge both about MACSYMA and about mathematics.

5.10 Some RALCMs for Mechanics

```
;This RALCM knows the velocity and position of the object
;moving on the surface, it is however completely
;ignorant of the possibility of flying off.
(DEFINE-RALCM  MASS-MOVEMENT  (?OBJECT ?SURFACE ?T1 ?T2)
                              ((A (ACCELERATION ?OBJECT)))
          (EVAL (?OBJECT ?T2 @POINT)
                (FETCH1 "(AT ,?OBJECT !>@POINT ,?T2)))
          (FCOND ((PRESENT (CONCAVITY ?SURFACE ZERO))
                  ;if the surface is flat, try simple kinematics
                  (FPROG ((THETA (ANGLE1 ?SURFACE) ACUTE))
                         (RALCM RTRI (?SURFACE))
                         (RALCM KIN (?OBJECT ?SURFACE ?T1 ?T2))
                         (FCOND ((PRESENT (HIGHER ?SURFACE !>NIL @POINT))
                                 (EQUATION
                                   (= A (* $ G (SIN THETA)))))
                                ((PRESENT (HIGHER ?SURFACE @POINT !>NIL))
                                 (EQUATION
                                   (= A (* -1 $ G (SIN THETA)))))))))
          (FCOND ((EVAL (@EXPERT @TYPE @POINT ?SURFACE ?T2 ?OBJECT)
                        (FETCH1 "(TYPE ,?SURFACE !>@TYPE))
                        (FETCH1 "(AT ,?OBJECT !>@POINT ,?T2))
                        (SETQ @EXPERT (GET @TYPE 'PARAMETEREXPERT)))
                  (RALCM @EXPERT (?SURFACE @POINT))))
          (RALCM ENERGY (?OBJECT ?SURFACE ?T1 ?T2)))
          ;energy will work for arbitrary shapes


;This RALCM knows about the principle of conservation of energy
;from a global perspective.
(DEFINE-RALCM GLOBAL-ENERGY (?OBJECT ?T1 ?T2)
                            ((VI (VELOCITY ?OBJECT ?T1) POSITIVE)
                             (VF (VELOCITY ?OBJECT ?T2) POSITIVE))
          (EVAL (@PT1 @PT2 ?OBJECT ?T1 ?T2)
                (FETCH1 "(AT ,?OBJECT !>@PT1 ,?T1))
```

```
                    (FETCH1 "(AT ,?OBJECT !>ePT2 ,?T2)))
                (FCOND ((ORDER ePT1 ePT2)
                        (FPROG ((H (HEIGHT ePT1 ePT2)))
                               (EQUATION
                                  (= (* VF VF) (+ (* VI VI) (* $ G -2 H))))))
                       ((ORDER ePT2 ePT1)
                        (FPROG ((H (HEIGHT ?PT2 ?PT1)))
                               (EQUATION
                                  (= (* VF VF) (+ (* VI VI) (* $ G 2 H))))))))

;This RALCM references knowledge about the local geometric relationships.
(DEFINE-RALCM LOCALHEIGHT (?SURFACE)
                          NIL
              (FCOND ((PRESENT (CONCAVITY ?SURFACE ZERO))
                      (RALCM RTRI (?SURFACE)))
                     ((EVAL (eEXPERT eTYPE ePOINT ?SURFACE)
                            (FETCH1 "(TYPE ,?SURFACE !>eTYPE))
                            (FETCH1 "(HIGHER ,?SURFACE !>ePOINT !>NIL))
                            (SETQ eEXPERT (GET eTYPE 'PARAMETEREXPERT)))
                      (RALCM eEXPERT (?SURFACE ePOINT)))))

;This RALCM knows about the principle of conservation of energy
;from a local perspective.
(DEFINE-RALCM ENERGY (?OBJECT ?SURFACE ?T1 ?T2)
                     ((VI (VELOCITY ?OBJECT ?T1) POSITIVE)
                      (VF (VELOCITY ?OBJECT ?T2) POSITIVE)
                      (H (HEIGHT ?SURFACE) POSITIVE))
              (EVAL (ePT1 ePT2 ?OBJECT ?T1 ?T2)
                    (FETCH1 "(AT ,?OBJECT !>ePT1 ,?T1))
                    (FETCH1 "(AT ,?OBJECT !>ePT2 ,?T2)))
              (FCOND ((PRESENT (HIGHER ?SURFACE ePT1 ePT2))
                      (EQUATION (= (* VF VF) (+ (* VI VI) (* 2 $ G H)))))
                     ((PRESENT (HIGHER ?SURFACE ePT2 ePT1))
                      (EQUATION (= (* VF VF) (* VI VI) (* -2 $ G H)))))))


;This RALCM knows about the relationships that hold within a right triangle.
(DEFINE-RALCM  RTRI (?TRIANGLE)
                    ((H (HEIGHT ?TRIANGLE) POSITIVE)
                     (L (BASE ?TRIANGLE) POSITIVE)
                     (HYP (DISTANCE ?TRIANGLE) POSITIVE)
                     (T1 (ANGLE1 ?TRIANGLE) ACUTE)
                     (T2 (ANGLE2 ?TRIANGLE) ACUTE))

             (INDEPENDENT 2
                 (EQUATION (= (+ T1 T2) (// $ pi 2)))
                 (EQUATION (= (TAN T1) (// H L)))
                           ; definition of tangent
                 (EQUATION (= (TAN T2) (// L H)))
                           ; definition of tangent
                 (EQUATION (=  HYP (SQRT (+ (* H H) (* L L)))))
                           ; pythagorean theorem
                 (EQUATION (= (SIN T1) (// H HYP)))
```

```
                        ; definition of sine
                (EQUATION (= (SIN T2) (// L HYP)))))
                        ; definition of sine

;This RALCM contains the three well known kinematic equations
(DEFINE-RALCM  KIN (?OBJECT ?SURFACE ?T1 ?T2)
                   ((VF (VELOCITY ?OBJECT ?T2))
                    (VI (VELOCITY ?OBJECT ?T1) POSITIVE)
                    (D (DISTANCE ?SURFACE))
                    (T (TIME ?T1 ?T2) POSITIVE)
                    (A (ACCELERATION ?OBJECT)))
                (INDEPENDENT 2
                    (EQUATION (= VF (+ VI (* A T))))
                    (EQUATION (= (* VF VF) (+ (* VI VI) (* 2 A D))))
                    (EQUATION (= D (+ (* VI T) (* .5 A T T))))))


;ZERO-POINT tries to calculate the point
;at which the force on the object becomes zero and leaves the surface
;?T2 is a hypothetical time and not necessary equivalent to
;end of ?SURFACE
(DEFINE-RALCM ZERO-POINT (?OBJECT ?SURFACE ?T1 ?T2)
                          ((NORMALF (FORCE ?OBJECT ?SURFACE ?T2)))
            (EVAL (eSPLITPOINT eSURFACE ePOINT ?OBJECT ?T1 ?T2)
                  (FETCH1 "(AT ,?OBJECT !>eSPLITPOINT ,?T2))
                  (FETCH1 "(AT ,?OBJECT !>ePOINT ,?T1))
                  (FETCH1 "(!>eSURFACE SEGMENT ,ePOINT ,eSPLITPOINT)))
            (EQUATION (= NORMALF 0))
                  (RALCM SPLIT-SURFACE (?SURFACE eSPLITPOINT))
                  (RALCM NORMALF (?OBJECT ?SURFACE ?T2))
                  (RALCM ENERGY (?OBJECT eSURFACE ?T1 ?T2)))


;This RALCM knows how to set up an equation
;for the normal force.
(DEFINE-RALCM NORMALF (?OBJECT ?SURFACE ?TIME)
                      ((V (VELOCITY ?OBJECT ?TIME) POSITIVE)
                       (NORMALF (FORCE ?OBJECT ?SURFACE ?TIME)))
             (EVAL (eEXPERT eTYPE ePOINT ?SURFACE ?TIME ?OBJECT)
                   (FETCH1 "(AT ,?OBJECT !>ePOINT ,?TIME))
                   (FETCH1 "(TYPE ,?SURFACE !>eTYPE))
            (SETQ eEXPERT (GET eTYPE 'PARAMETEREXPERT)))
            (RALCM eEXPERT (?SURFACE ePOINT))
            (FPROG ((THETA (ANGLE ?SURFACE ePOINT) ACUTE)
                    (R (CURVATURE ?SURFACE ePOINT) POSITIVE))
                   (EQUATION
                      (= NORMALF
                         (+ (// (* -1 V V) R) (* $ G (COS THETA)))))))

;This RALCM is the canonical parameterization for a circle
(DEFPROP CIRCLE CIRCLE PARAMETEREXPERT)
;The ?POINT can be an endpoint of the ?SURFACE or some other point
```

```
;in between the endpoints.
(DEFINE-RALCM CIRCLE (?SURFACE ?POINT)
                     ((CURVATURE (CURVATURE ?SURFACE ?POINT) POSITIVE)
                      (RADIUS (RADIUS ?SURFACE) POSITIVE)
                      (THETA2 (ANGLE ?SURFACE ?POINT) ACUTE))
     ;The radius of curvature for a circle is fixed and equal to the radius
     (EQUATION (= RADIUS CURVATURE))
     ;All paramterizations have their initial point at the lowest point
     (EVAL (@SEGMENT @START ?POINT ?SURFACE)
                            (FETCH1 "(HIGHER ,?SURFACE !>NIL !>@START))
                            (OR
                              (FETCH1 "(HIGHER !>@SEGMENT ,?POINT ,@START))
                              (SETQ @SEGMENT ?SURFACE)))
     ;Using this initial point, other quantities can be determined.
     (FPROG ((THETA1 (ANGLE ?SURFACE @START) ACUTE)
             (HEIGHT (HEIGHT @SEGMENT) POSITIVE))
     ;Since heights are always positive.
           (FCOND ((PRESENT (CONCAVITY ?SURFACE PLUS))
                   (EQUATION
                      (= HEIGHT
                         (* RADIUS (+ (COS THETA1) (* -1 (COS THETA2))))))
                  ((PRESENT (CONCAVITY ?SURFACE MINUS))
                   (EQUATION
                      (= HEIGHT
                         (* RADIUS (+ (COS THETA2) (* -1 (COS THETA1)))))))))))
;SPLIT-SURFACE sets up some of the relations that
;need to be present when a segment is split.
(DEFINE-RALCM SPLIT-SURFACE (?SURFACE ?SPLITPOINT)
                            ((HS (HEIGHT ?SURFACE) POSITIVE))
              (EVAL (@BOTTOM @TOP @SURFACE1 @SURFACE2 ?SURFACE ?SPLITPOINT)
                    (FETCH1 "(HIGHER ,?SURFACE !>@TOP !>@BOTTOM))
                    (FETCH1 "(HIGHER !>@SURFACE1 ,@TOP ,?SPLITPOINT))
                    (FETCH1 "(HIGHER !>@SURFACE2 ,?SPLITPOINT ,@BOTTOM)))
              (FPROG ((HS1 (HEIGHT @SURFACE1) POSITIVE)
                      (HS2 (HEIGHT @SURFACE2) POSITIVE))
                     (EQUATION (= HS (+ HS1 HS2))))))
```

5.11 Critique

Many of the previous sections of this chapter can be considered as criticisms of the original theory in that they described patches to the theory rather than a basic solution. Notwithstanding these criticisms and those that follow, the organization as outlined does actually function correctly. The criticism really is that the way it functions is fundamentally incorrect and that this leads to problems of inefficiency and communication with other knowledge.

Once a collection of RALCMs has been used to solve a problem the FCONDs that were met

can be used as tests to detect whether the same problem is being solved again. Thus a problem type, once solved, need never be solved again. There is no fundamental reason this was not done in NEWTON other than that this research was not interested in this simple kind of learning. The observation does, however, point to a more interesting problem.

Suppose a RALCM is invoked with all its secondary variables unknown. This RALCM will return a general complaint involving all possible variables. Once such a complaint has been generated the RALCM need never again be searched. Invoking the RALCM on different purposes merely means that different top nodes will be returned, but the graph this node points to can always be the same. Every time the RALCM is invoked this prototype complaint can be simplified according to the variables that are currently known and the resultant complaint or success indication returned. The general complaint expresses the dependencies between the equations and variables of the RALCM in a graph form. Hence NEWTON should compile the RALCMs into the general prototype complaint when the RALCM is read in and this graph searched instead of the original awkward linear formulation of the dependencies.

Suppose RTRI were invoked without having any secondary variables known. Currently IRALCM would search through all of its equations ignoring the well known fact that at least three quantities about a triangle need to be known before it is uniquely determined. The compilation of the previous paragraph partially resolves this problem in that a complaint is immediately returned asking for any of the other variables. It still, however, does not contain the fact that three facts about a triangle have to be known before it can be solved for.

With such modifications simultaneity and redundancy become problems; however, they need re-examination anyway. The way IRALCM is currently structured it checks for simultaneity only as a last resort. These checks should take place while the problem is being solved, but that would introduce incredible inefficiency in the solving process.

All these problems are symptoms of the basic flaw in IRALCM. This thesis has advocated planning, yet once a problem is posed to IRALCM no more planning is done. Basically we have built a routine which takes a set of equations and grinds them within a black box isolated from the

rest of NEWTON. It is not surprising then that IRALCM encounters many difficulties. IRALCM should plan using the qualitative meaning of the equations and not use some uniform procedure to grind equations.

6.0 NEWTON SCENARIOS

6.1 Posing the Question

This chapter consists of a series of scenarios of NEWTON solving problems. Among those problems solved are the two examples from chapter 2.

The problem is posed to NEWTON through a DEFINE-PROBLEM statement.

```
(DEFINE-PROBLEM name form1 form2 ...)
```

The name gives the problem a tag by which it can be referenced. The subsequent forms either provide information about the problem or are requests to answer questions.

One major part of the problem presentation is the description of the scene for the envisioner. This is given by the SCENE form:

```
(SCENE descriptor1 descriptor2 ...)
```

The descriptors are those developed for SEGMENT and CORNER in the chapter on envisioning. A segment is described:

```
(name SEGMENT highpoint lowpoint type concavity parity)
```

The parity has no theoretical significance, NEWTON uses it to keep track of right and left internally. A corner is described by:

```
(name CORNER point1 point2 slopechange)
```

The scene for the sliding block problem is described:

```
(DEFINE-PROBLEM SLIDING-BLOCK
     (SCENE
          (C1 CORNER * S1 -1)
          (S1 SEGMENT C1 C2 INCLINE PLUS 1)
          (C2 CORNER S1 S2 0)
          (S2 SEGMENT C3 C2 INCLINE PLUS 1)
          (C3 CORNER  S2 S3 0)
          (S3 SEGMENT C4 C3 INCLINE ZERO 1)
          (C4 CORNER  S3 * -1))
     ...)
```

To place the object in the scene the PLACE form is used:

```
(PLACE location contact position direction)
```

In our example:

```
(PLACE C1 ON ABOVE Z)
```

To give values to a variable the ASSIGN form is used:

```
(ASSIGN variable value)
```

In the particular version of the sliding block problem presented here the object will make it to the straight section but will be unable to traverse it:

```
(ASSIGN (HEIGHT S1) 2)
(ASSIGN (HEIGHT S2) 1)
(ASSIGN (BASE S3) 2)
(ASSIGN (ANGLE1 S3) 1)
```

The QUERY form can be used to ask qualitative questions. To ask whether the object makes it to a particular position:

```
(QUERY (REACH C4))
```

These are the basic forms used to pose the problem, more will be introduced as new problems are discussed.

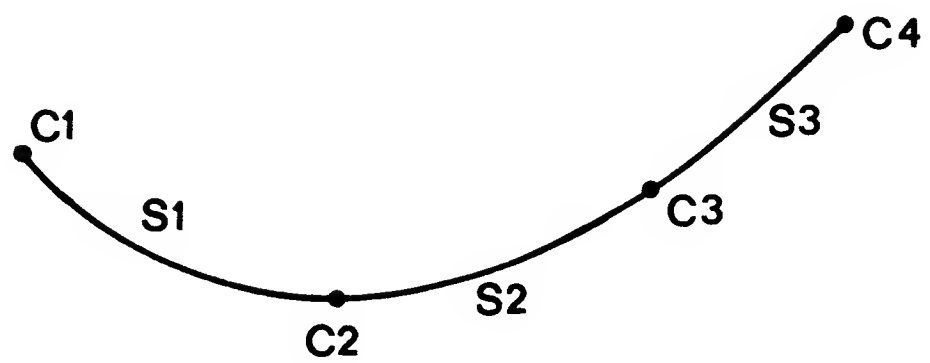
6.2 The Sliding Block Problem

The previous section outlined the complete presentation of the problem:

```
(DEFINE-PROBLEM SLIDING-BLOCK
     (SCENE
          (C1 CORNER * S1 -1)
          (S1 SEGMENT C1 C2 INCLINE PLUS 1)
          (C2 CORNER S1 S2 0)
          (S2 SEGMENT C3 C2 INCLINE PLUS 1)
          (C3 CORNER  S2 S3 0)
          (S3 SEGMENT C4 C3 INCLINE ZERO 1)
          (C4 CORNER  S3 * -1))
     (PLACE C1 ON ABOVE Z)
     (ASSIGN (HEIGHT S1) 2)
     (ASSIGN (HEIGHT S2) 1)
     (ASSIGN (BASE S3) 2)
     (ASSIGN (ANGLE1 S3) 1)
     (QUERY (REACH C4))
```

If NEWTON was explicitly told that the point C4 was higher than the point C1 the envisioning would deduce that C4 was unreachable. Let us assume that this information, which might have been deducible from the diagram is not given.

We have already seen the envisionment for this problem. As a reference, the diagram will be given again.



The initial unpruned plan implied by this envisioning is a SLIDE-SLIDE ambiguity on S2 and S3. A global expert infers that the first step of the plan is unnecessary. NEWTON then invokes the QA for a SLIDE-SLIDE ambiguity on S3. This QA instantiates the MASS-MOVEMENT RALCM with a request for the final velocity, this results in instantiations of ENERGY, KIN and RTRI. The QA fails and complains back. One of the complaints is that the initial velocity on S3 is unknown and the QA now tries to solve for that variable. Because the initial velocity at C1 is known, the global energy strategy which requires the height difference between C1 and C3 is attempted. This height is easily computed from the givens, the original MASS-MOVEMENT RALCM returns with a contradiction and this results in the global contradiction that C4 is not reachable.

NEWTON prints out a trace of its analysis as it solves the problem. From the discussion of the previous paragraph this trace should be easily interpretable and explanations will be added in italics wherever necessary.

First the envistonment is done and the assignment of times to points is added to the data base:

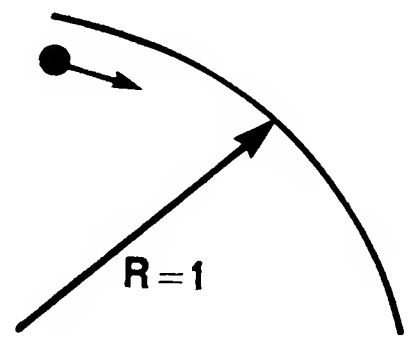
```
(AT X C1 T71)
(AT X S1 T70)
(AT X C2 T69)
(AT X S2 T68)
(AT X C3 T67)
(AT X S3 T66)
(AT X C4 T65)
Attempting to determine if C4 is reachable; plan:
S2 QA-SLIDE-SLIDE
```

```
S3 QA-SLIDE-SLIDE
Deleting plan step: QA-SLIDE-SLIDE S2
Global height expert invoked for HEIGHT-C1-C4
```
The global height expert constructs a RALCM which it will instantiate:
```
DEFINE-RALCM: HEIGHTS
DEFINE-RALCM: BASICHEIGHTS
Instantiating HEIGHTS on NIL
Instantiating BASICHEIGHTS on NIL
1 plans for HEIGHT-C1-C4 discovered.
PLAN:HEIGHT-S3
Attempting plan # 1
```
A RALCM is being invoked from a QA:
```
Attempting to find a value for (HEIGHT S3) in LOCALHEIGHT (S3)
Instantiating LOCALHEIGHT on (S3)
Instantiating RTRI on (S3)
```
IRALCM finds a path to a variable:
```
Solution found for  HEIGHT-S3
```
Note that HEIGHT-S3 *and* (HEIGHT S3) *are equivalent.*
```
Solution found for  HEIGHT-C3-GROUND
Solution found for  HEIGHT-C2-GROUND
Solution found for  HEIGHT-C1-GROUND
Solution found for  HEIGHT-C1-C4
```
IRALCM does not start solving the equations until it has found a complete path to the goal.
```
Equation B153 from RALCM RTRI ->

HEIGHT-S3 = 2 TAN(1)
```
The RALCMs that NEWTON uses do not have names on their EQUATIONS so IRALCM has created names such as B153 for them.
```
Equation B208 from RALCM BASICHEIGHTS ->

HEIGHT-C3-GROUND = HEIGHT-C4-GROUND - 2 TAN(1)

Equation B209 from RALCM BASICHEIGHTS ->

HEIGHT-C2-GROUND = HEIGHT-C4-GROUND - 2 TAN(1) - 1

Equation B210 from RALCM BASICHEIGHTS ->

HEIGHT-C1-GROUND = HEIGHT-C4-GROUND - 2 TAN(1) + 1

Equation B203 from RALCM HEIGHTS ->

HEIGHT-C1-C4 = 2 TAN(1) - 1
```
Now that the global height difference has been determined the principle of conservation of energy can be used to calculate the velocity.
```
Attempting to find a value for (VELOCITY X T65)
                               in GLOBAL-ENERGY (X T71 T65)
Instantiating GLOBAL-ENERGY on (X T71 T65)
Solution found for  VELOCITY-X-T65
```

```
Equation B126 from RALCM GLOBAL-ENERGY ->
Ranges on variable VELOCITY-X-T65 result in contradiction
The velocity has been determined to be imaginary and that
is a contradictory value for a velocity.
The object fails to make the desired choice on or before S3
```

6.3 The Loop-the-loop Problem

Before discussing the loop-the-loop problem which was first presented in chapter 2, let us examine how a simpler problem might be posed to NEWTON.



An object moving at velocity $v = g$ along the underside of a circular surface as indicated in the diagram. The surface has a unit radius, will the object fall off?

NEWTON must be presented the description of the circular section. The information in the SCENE form remains unchanged, however the ASSERT form is used to communicate facts such as whether the surface is circular or not.

```
(ASSERT datum)
```

NEWTON has knowledge at various levels about some standard kinds of two dimensional curves. Currently NEWTON understands circles and orthogonally oriented parabolas. All NEWTON needs to understand a curve is its RALCM and the problem statement can easily include a RALCM definition to describe any given curve. The fact that a curve is of a type about which NEWTON knows something about is communicated by asserting (TYPE surface type) in an ASSERT form. To indicate that the surface S is a circle:

```
(ASSERT (TYPE S CIRCLE))
```

Two important facts about how NEWTON represents certain variables are necessary to be able to give it quantitative information about shapes. The tangent angle of a curve is given by an ANGLE variable. The angle must always be acute, being the positive acute angle between the

tangent and the horizontal. If no such angle is formed, it is assumed to be 0. Information provided within the SCENE form for that section of curve can be used to determine the angle of the tangent in some absolute coordinate scheme (e.g. if one wants to distinguish between positive or negative or measure clockwise rotation from positive horizontal axis). Usually an ANGLE variable references the point at which the angle is measured and the name of the surface it is a part of:

```
(ANGLE surface point)
```

For a flat segment this angle is constant so the point specification can be ignored:

```
(ANGLE surface)
```

Within NEWTON heights are always positive. To determine the difference in heights between two points the HEIGHT variable gives the magnitude of this height difference and assertions in the data base (usually added by the SCENE form) are used to determine the correct sign of this quantity. HEIGHT variables have three different formats.

```
(HEIGHT surface)
```

This gives the height difference between the top and bottom points of the surface.

```
(HEIGHT surface point)
```

This gives the height from the bottom of the surface to the given point.

```
(HEIGHT point1 point2)
```

This gives the magnitude of the height between the two points. These forms are never used interchangeably; the form with the least number of references to points is always used. The latter form is the most general, but if the two points lie on the same segment and one point was an endpoint one of the other two forms are used. If both points are endpoints the first form is used.

The simple problem we have been discussing would be posed as follows:

```
(DEFINE-PROBLEM  SIMPLE-CIRCLE
      (SCENE (C1 CORNER * S1 -1)
             (S1 SEGMENT C1 C2 INCLINE NEGATIVE 1)
             (C2 CORNER S2 * -1))
      (PLACE C1 ON BELOW 0)
      (SEQUENCE (?INITIAL C1))
      (ASSERT (TYPE S1 CIRCLE))
      (ASSIGN (ANGLE S1 C1) 0)
      (ASSIGN (ANGLE S1 C2) $ PI/2)
```

```
(ASSIGN (RADIUS S1) 1)
(ASSIGN (VELOCITY X ?INITIAL) $ G)
(QUERY (FALL-OFF)))
```

This simple instance of the SEQUENCE form binds a temporary variable so that the VELOCITY variable which references an as yet unknown time variable can be given a value. Later we shall see more sophisticated uses of this form.

To describe a scene like that for a roller coaster a large number of assertions have to be made to describe just the circle. The radius for each section must be individually given, the angle at each end of each segment must be given and each segment must be explicitly given a CIRCLE type. In order to express the problem in a more convenient way NEWTON contains a procedure for automatically compiling the assertions for a circle. Just as there is a primitive for SEGMENT and CORNER there is a primitive SCENE descriptor called CIRCLE. It is important to note that the CIRCLE primitive merely expands into its constituent SEGMENTs and CORNERs adding the correct assertions to the data base. The circle primitive for the SCENE form is given as:

```
(name CIRCLE angle1 corner1
             angle2 corner2
             concavity1 clockwise parity points radius)
```

The angle and corner parameters describe the initial and final corners and the tangents at those points. The concavity1 indicates that initial concavity of the curve. The clockwise flag indicates whether the circle is drawn clockwise or counterclockwise from its starting point. The parity has its usual meaning. Points gives the number of quadrants of the circle that will have to be generated and the radius gives the radius for the entire circle. The description of the simple circle problem we have been considering can now be reduced to:

```
(DEFINE-PROBLEM  SIMPLE-CIRCLE
      (SCENE (C CIRCLE 0        (C1 CORNER * ?S -1)
                     $ PI/2   (C2 CORNER ?S * -1)
                     NEGATIVE T 1 1 1.0))
      (PLACE C1 ON BELOW O)
      (SEQUENCE (?INITIAL C1))
      (ASSIGN (VELOCITY X ?INITIAL) $ G)
      (QUERY (FALL-OFF)))
```

The ?S variable indicates a slot that must be filled in by the CIRCLE macro as it creates the names

of the segments that make up the circle.

We are now ready to see how the loop-the-loop problem could be posed:

```
(DEFINE-PROBLEM LOOP-THE-LOOP
      (SCENE (C1 CORNER * S1 -1)
             (S1 SEGMENT C0 C1 INCLINE PLUS 1)
             (C  CIRCLE 0 (C2 CORNER S1 ?S 0)
                          0 (X  CORNER ?S * -1)
                          POSITIVE NIL 1 4 R))
      (PLACE C1 ON ABOVE Z)
      (ASSIGN (HEIGHT S1) *)
      (QUERY (REACH X)))
```

The radius parameter of the CIRCLE primitive expands into a sequence of ASSIGN forms for the radius. The value parameter for the ASSIGN form need not be a number. Any expression appearing as the value is assumed to be a constant. So that in (ASSIGN (RADIUS C) R) the R is assumed to be a constant and NEWTON will never attempt to obtain a value for it unless explicitly asked to. The only possible advantage of using such a construct is to inform NEWTON that two (perhaps separated) circular segments have the same radius. That fact may result in substantial simplifications in the problem. The value can also be '*' indicating that this is a boundary condition which must be resolved for. When NEWTON interprets the later QUERY form any attempt to disambiguate between goals which fails because of '*' variables being unknown is assumed to have a positive outcome. The constraint that the positive outcome has on the '*' variables is remembered as it is the constraints themselves which are the most interesting in such a problem.

The following is a trace of NEWTON solving the problem. Refer to the trace for the sliding block problem for more details.

```
(AT X C1 T100)
(AT X S1 T99)
(AT X C2 T98)
(AT X S33 T97)
(AT X P25 T96)
(AT X S34 T95)
(AT X P26 T94)
(AT X S35 T93)
(AT X P27 T92)
(AT X S36 T91)
(AT X C6 T90)
```

```
Attempting to determine if C6 is reachable; plan:
S33 QA-SLIDE-SLIDE
S34 QA-FALL-SLIDE
S35 QA-FALL-SLIDE
Deleting plan step: QA-SLIDE-SLIDE S33
Deleting plan step: QA-FALL-SLIDE S35
Attempting to find a value for (FORCE X S34 T94) in NORMALF (X S34 T94)
Instantiating NORMALF on (X S34 T94)
Instantiating CIRCLE on (S34 P26)
Solution found for  CURVATURE-S34-P26
```

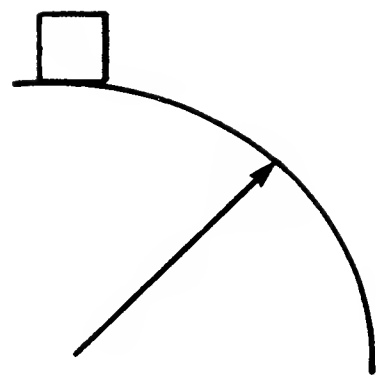
NEWTON realizes that the velocity is required and it tries to compute the global height. Note that NEWTON did not make any inferences about the height of the circle when it read the problem.

```
DEFINE-RALCM: HEIGHTS
DEFINE-RALCM: BASICHEIGHTS
Instantiating HEIGHTS on NIL
Instantiating BASICHEIGHTS on NIL
1 plans for HEIGHT-C1-P26 discovered.
PLAN:HEIGHT-S34 HEIGHT-S33
Attempting plan # 1
Attempting to find a value for (HEIGHT S34) in LOCALHEIGHT (S34)
Instantiating LOCALHEIGHT on (S34)
Solution found for  HEIGHT-S34
Solution found for  HEIGHT-P25-GROUND
Equation B85 from RALCM CIRCLE ->

HEIGHT-S34 = R

Attempting to find a value for (HEIGHT S33) in LOCALHEIGHT (S33)
Instantiating LOCALHEIGHT on (S33)
Instantiating CIRCLE on (S33 P25)
Solution found for  HEIGHT-S33
Solution found for  HEIGHT-C2-GROUND
Solution found for  HEIGHT-C1-GROUND
Solution found for  HEIGHT-C1-P26
Equation B83 from RALCM CIRCLE ->

HEIGHT-S33 = R

Equation B245 from RALCM BASICHEIGHTS ->

HEIGHT-P25-GROUND = HEIGHT-P26-GROUND - R

Equation B246 from RALCM BASICHEIGHTS ->

HEIGHT-C2-GROUND = HEIGHT-P26-GROUND - 2 R

Equation B247 from RALCM BASICHEIGHTS ->

HEIGHT-C1-GROUND = - 2 R + HEIGHT-S1 + HEIGHT-P26-GROUND

Equation B238 from RALCM HEIGHTS ->
```

```
HEIGHT-C1-P26 = 2 R - HEIGHT-S1

Attempting to find a value for (VELOCITY X T94)
                                in GLOBAL-ENERGY (X T100 T94)
Instantiating GLOBAL-ENERGY on (X T100 T94)
Solution found for  VELOCITY-X-T94
```
When the velocity is discovered IRALCM instantly realizes that a path to the destred force has been discovered.
```
Solution found for  FORCE-X-S34-T94
Equation B27 from RALCM GLOBAL-ENERGY ->

VELOCITY-X-T94 = SQRT(2) SQRT( G) SQRT(HEIGHT-S1 - 2 R)

Equation B77 from RALCM CIRCLE ->

CURVATURE-S34-P26 = R

Equation B74 from RALCM NORMALF ->

                          2  G (HEIGHT-S1 - 2 R)
FORCE-X-S34-T94 =  G - ----------------------
                                  R

Constraint on FORCE-X-S34-T94 implies that
                    5 R
(HEIGHT-S1 - ---) > 0
                    2

globalconstraints:

                    5 R
(HEIGHT-S1 - ---) > 0
                    2
```

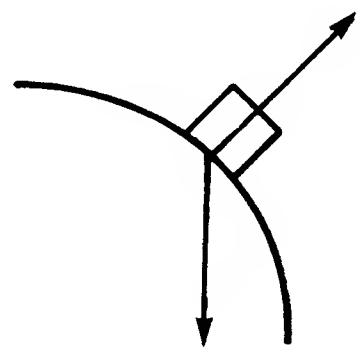
6.4 The Great Dome Problem

We have, as yet, not discussed this problem. The great dome problem will be first worked out so that the trace of NEWTON's working out the problem can be better understood.



A small block slides from rest from the top of a frictionless sphere of radius r. How far below the top does it lose contact with the sphere?

The block will slide downwards. At the very start of its motion it will remain in contact with the sphere since the surface is horizontal at the top. If the block ever did reach the equator it would have to immediately fall off as it would start sliding underneath the sphere. So the block loses contact with the sphere somewhere between the pole and the equator. At any point the block will either continue sliding along the surface or fly off, we know it will fly off before it reaches the equator, but where is that point? The forces on the block must be examined. There are only two forces acting on the block, one is gravity and the other is the reaction of the sphere.



To remain in contact with the sphere the component of the total force in the direction of the center of the sphere must be sufficient for the block to continue in circular motion:

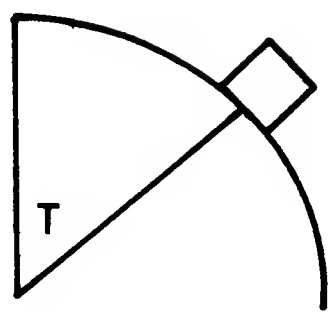
$$m v^2 / r$$

Since the direction of the reaction of the sphere is normal to its surface and thus away from its center we can ignore that force and just solve for the point at which the component of the force of gravity in the direction of the center is insufficient to maintain circular motion. This happens

when:

$$m v^2 / r = m g \cos T$$

Where the angle T is as defined in the diagram:



The velocity of the block can be computed by conservation of energy from the distance the block has already dropped through:

$$1 / 2 m v^2 = m g r (1-\cos T)$$

These two equations can be easily solved for T:

$$\cos T = 2 / 3.$$

This problem would be posed:

```
(DEFINE-PROBLEM GREAT-DOME
       (SCENE (C CIRCLE 0 (C1 CORNER * ?S 0)
                          (C2 CORNER ?S * -1)
                          NEGATIVE T 1 1 R))
       (PLACE C1 ON ABOVE Z)
       (QUERY (FLY-OFF) (POINT ?POINT)  (SURFACE ?SURFACE))
       (QUERY (ANGLE ?POINT ?SURFACE)))
```

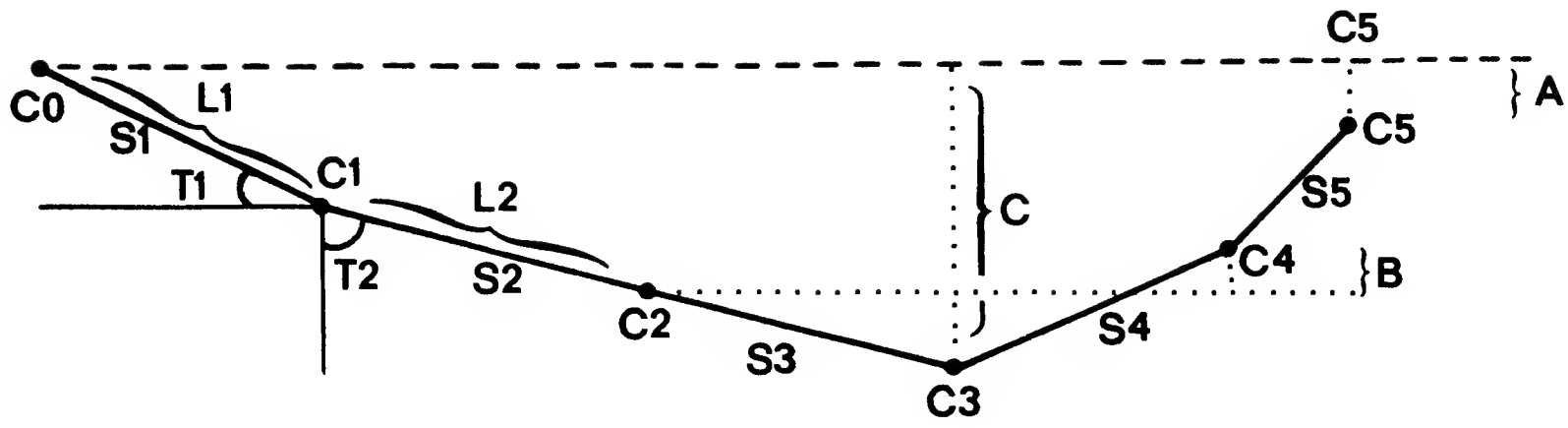
Following the aspect request of the QUERY form a sequence of pairs indicating variables of the aspect to be bound to temporary variables so that quantitative variables about that aspect can be referenced.

```
(AT X C1 T74)
(AT X S46 T73)
To solve this problem a hypothetical point at which
a object leaves the surface is created and its exact position calculated.
Creating hypothetical point P16 which divides S46 into S47 and S48
(AT X P16 T75)
```

```
Attempting to find a value for (ANGLE S46 P16)
                                in ZERO-POINT (X S46 T74 T75)
Instantiating ZERO-POINT on (X S46 T74 T75)
Instantiating SPLIT-SURFACE on (S46 P16)
Instantiating NORMALF on (X S46 T75)
Instantiating CIRCLE on (S46 P16)
Instantiating ENERGY on (X S47 T74 T75)
Solution found for  CURVATURE-S46-P16
Solution found for  FORCE-X-S46-T75
```
IRALCM now detects that the equations are simultaneous in the angle.
So it hypothesizes ANGLE-S46-P16 as a parameter.
```
SIMULTANEITY: 1 subgraphs.
Equation B184 from RALCM CIRCLE ->

HEIGHT-S48 = COS(ANGLE-S46-P16) R

Equation B190 from RALCM SPLIT-SURFACE ->

HEIGHT-S47 = R - COS(ANGLE-S46-P16) R

Equation B146 from RALCM ENERGY ->

VELOCITY-X-T75 = SQRT(2) SQRT( G) SQRT(R - COS(ANGLE-S46-P16) R)

Equation B176 from RALCM CIRCLE ->

CURVATURE-S46-P16 = R

Equation B165 from RALCM ZERO-POINT ->

FORCE-X-S46-T75 = 0
```
Substituting, the angle is eliminated.
```
Equation B173 from RALCM NORMALF ->

                                2
ANGLE-S46-P16 = ACOS(-)
                                3
```

6.5 A Geometry Problem

NEWTON is primarily designed to solve problems in simple mechanics. Such problems often involve subproblems about the geometry of the scene. In chapter 5, a procedure was outlined for planning and determining the height between two points. The following is a simple problem about heights.



If the indicated heights are known, what is the height between C2 and C5.

```
(DEFINE-PROBLEM HEIGHTS
       (SCENE
           (C0 CORNER * S1 0)
           (S1 SEGMENT C0 C1 INCLINE ZERO 1)
           (C1 CORNER  S1 S2 0)
           (S2 SEGMENT C1 C2 INCLINE ZERO 1)
           (C2 CORNER  S2 S3 0)
           (S3 SEGMENT C2 C3 INCLINE ZERO 1)
           (C3 CORNER  S3 S4 0)
           (S4 SEGMENT C4 C3 INCLINE ZERO 1)
           (C4 CORNER  S4 S5 0)
           (S5 SEGMENT C5 C4 INCLINE ZERO 1)
           (C5 CORNER  S5 * 0))
      (ASSIGN (HEIGHT C0 C5) A)
      (ASSIGN (HEIGHT C2 C4) B)
      (ASSIGN (HEIGHT C3 C5) C)
      (ASSIGN (DISTANCE S1) L1)
      (ASSIGN (DISTANCE S2) L2)
      (ASSIGN (ANGLE1 S1) T1)
      (ASSIGN (ANGLE2 S2) T2)
      (QUERY (HEIGHT C2 C5)))
```

```
Global height expert invoked for HEIGHT-C2-C5
DEFINE-RALCM: HEIGHTS
DEFINE-RALCM: BASICHEIGHTS
Instantiating HEIGHTS on NIL
Instantiating BASICHEIGHTS on NIL
Solution found for  HEIGHT-C3-GROUND
Solution found for  HEIGHT-C0-GROUND
```

In this example IRALCM will attempt 3 bad plans before it succeeds.

```
4 plans for HEIGHT-C2-C5 discovered.
PLAN:HEIGHT-S3
PLAN:HEIGHT-S5
PLAN:HEIGHT-S4
PLAN:HEIGHT-S2 HEIGHT-S1
Attempting plan # 1
Attempting to find a value for (HEIGHT S3) in LOCALHEIGHT (S3)
Instantiating LOCALHEIGHT on (S3)
Instantiating RTRI on (S3)
Plan fails on: HEIGHT-S3
Attempting plan # 2
Attempting to find a value for (HEIGHT S5) in LOCALHEIGHT (S5)
Instantiating LOCALHEIGHT on (S5)
Instantiating RTRI on (S5)
Plan fails on: HEIGHT-S5
Attempting plan # 3
Attempting to find a value for (HEIGHT S4) in LOCALHEIGHT (S4)
Instantiating LOCALHEIGHT on (S4)
Instantiating RTRI on (S4)
Plan fails on: HEIGHT-S4
Attempting plan # 4
Attempting to find a value for (HEIGHT S2) in LOCALHEIGHT (S2)
Instantiating LOCALHEIGHT on (S2)
Instantiating RTRI on (S2)
Solution found for  ANGLE1-S2
Solution found for  HEIGHT-S2
Solution found for  BASE-S2
Equation B152 from RALCM RTRI ->

                PI
ANGLE1-S2 = --- - T2
                2

Equation B156 from RALCM RTRI ->

HEIGHT-S2 = L2 COS(T2)

Attempting to find a value for (HEIGHT S1) in LOCALHEIGHT (S1)
Instantiating LOCALHEIGHT on (S1)
Instantiating RTRI on (S1)
Solution found for  HEIGHT-S1
Solution found for  HEIGHT-C1-GROUND
Solution found for  HEIGHT-C2-GROUND
Solution found for  HEIGHT-C2-C5
Solution found for  HEIGHT-C4-GROUND
Solution found for  HEIGHT-S3
Solution found for  HEIGHT-S5
Solution found for  HEIGHT-S4
Equation B156 from RALCM RTRI ->

HEIGHT-S1 = L1 SIN(T1)

Equation B220 from RALCM HEIGHTS ->
```

HEIGHT-C0-GROUND = HEIGHT-C5-GROUND - A

Equation B229 from RALCM BASICHEIGHTS ->

HEIGHT-C1-GROUND = - L1 SIN(T1) + HEIGHT-C5-GROUND - A

Equation B228 from RALCM BASICHEIGHTS ->

HEIGHT-C2-GROUND = - L2 COS(T2) - L1 SIN(T1) + HEIGHT-C5-GROUND - A

Equation B217 from RALCM HEIGHTS ->

HEIGHT-C2-C5 = L2 COS(T2) + L1 SIN(T1) + A

7.0 CONCLUDING REMARKS

7.1 Limitations

By this time the reader may have concluded that NEWTON is a true expert in the roller coaster world and that further research should involve a more complex world. In actual fact NEWTON understands very little about the roller coaster world. Two major limitations are the lack of forms of knowledge and an incomplete development of the forms of knowledge NEWTON does have. Limitations in the forms of knowledge NEWTON has available preclude the possibility of explicitly expressing plans. NEWTON's inability to envision rolling objects is an example of a lack of development of a form of knowledge NEWTON has available. In this section these limitations will be reiterated and viewed from a more global perspective. The next section discusses what generalizations are possible within the current framework. Following that section the chapter concludes with discussions about the generalizability of what has been discovered.

ENVISIONING LIMITATIONS:

The envisioner for NEWTON only knows about the roller coaster world. The envisioner can be generalized to be able to handle more sophisticated problems, but this generalization would be restricted to envisioning requiring only one focus of attention and requiring no consideration of volume.

RALCM LIMITATIONS:

Anything that can be encoded into a RALCM can be used by NEWTON. Any deficiency in quantitative knowledge can be easily rectified. However, certain kinds of problems cause excessive computation to take place. The reason for this is not so much that the RALCMs are an inadequate theory, but rather explicit planning must be done before giving the problem to IRALCM. An example of this was the necessity for an expert about heights.

QA LIMITATIONS:

In retrospect, it is the QAs that play the fundamental role in the problem solving. The QAs use the other knowledge to produce plans and subsequently control their execution. It is not surprising then that most of the problems of NEWTON eventually surface with the QAs. If other knowledge in NEWTON is insufficient the QAs will not be able to instantiate the plans and if the QAs themselves are deficient the plans will be unconstructable. In both cases the problem will seem to arise with the QAs.

The QAs themselves can be divided into different types. The reason that was not done in NEWTON was that the nature of these divisions is not yet well enough understood. The first division occurs between those QAs that deal with the statement of the problem directly to construct a plan and those that subsequently interpret that plan. The task of building the initial plan can also be divided into domain dependent and independent sections. The domain independent section analyzes the structure of the question to determine the kind of question was being asked and the domain dependent section instantiates the plan indicated by the problem type to the domain.

An example of domain independent planning is determining whether the boundary conditions were enforced or inquired about. A problem requesting the boundary conditions for a desired effect has a top level plan that is independent of any domain. A common top level plan is to introduce hypothetical boundary conditions and see what constraints the desired effects have on these hypotheses. This could be as much a plan for a mechanics problem as for an electronics problem. The point is that there is a great deal of expertise about analyzing questions and generating abstract plans for their solution that is completely independent of the domain the question is set within. Currently NEWTON does very little of this metaplanning and hence is incapable of handling many questions for which it contains sufficient knowledge to solve (if the question were posed in a different way).

More examples of such metaplanning are handling conjunctive and disjunctive goals, recognizing possible simultaneity and being able to delay finishing a plan until the problem has been further analyzed. The basic kind of plan that NEWTON handles is envision-QA-RALCMs.

Any variation or other interaction causes problems. NEWTON cannot handle interaction between the envisioning and other knowledge. The envisioner produces one envisionment for the problem and that is all. If more facts are discovered later that would affect the envisioner (e.g. discovering height information) the envisionment can not be modified. The necessity that some problems had for interaction between the QAs and the RALCMs was called indefiniteness in chapter 4. Indefiniteness is that characteristic which makes it difficult to construct a complete plan for the problem *a prtort*. For indefinite problems the plan must be developed as the problem unfolds.

MATHEMATICAL LIMITATIONS:

The mathematical expertise that NEWTON uses does not always succeed. A more serious shortcoming is that it does not know calculus. Although the mathematical expertise of handling the mathematical operators of differentiation and integration is well known, the interface between this mathematical expertise and the quantitative knowledge is a topic for research. The two principle points of difficulty are the introduction of unwanted redundancy and initial values. The introduction of integration and differentiation as operators means that many more equations are redundant (e.g. KIN would only have one independent equation). Integration is a very peculiar mathematical operator as it artificially introduces an unknown variable, namely the constant of integration.

Although most of NEWTON's problems manifest themselves in the QAs, the reason for the difficulty can lie as much with a lack of knowledge or flexibility in the knowledge as with the QAs. The following is a (partial) list of problems which NEWTON cannot solve. Each question annotated with a short comment explaining why NEWTON cannot handle it. Only from studying such examples will it become clear what role planning plays in problem solving. For those problems where a deficiency in the knowledge is the source of the difficulty, it will often be seen that it is not so much a lack of knowledge but the inflexibility of using the knowledge that NEWTON does have. Note that there are myriads of problems that NEWTON stands no hope of solving; the

questions that follow are problems about which NEWTON knows at least something and should be expected to know more.

"What is the period of an object on a cycloid?"
-- integration

"What is the velocity where the object flies off falls off?"
-- conjunctive goals

"What would be the answer to the loop-the-loop problem if the radius of the circle was equal to the velocity of the block at the bottom (in MKS units)?"
-- peculiar constraints

"Would the object still make it if the height of the hill was increased?"
-- asking about the effect of modifying a problem

"What is the curve of shortest time between two points?"
-- variational calculus

"What is the velocity on the first curve that is not circular?"
-- too indefinite

"An object is moving along a circular section whose change in angle can encompass one or two segments."
-- partial envisionment not possible

"Assuming that the object won't fly off what is the final velocity?"
-- inability to take advice

"If the fall-off velocity is determined what must the initial height be?"
-- quantitative constraints about hypothetical points

"The object loops around the loop-the-loop 100 times and then exits."
-- envisioner cannot understand such a dynamically changing surface

"What is the velocity of the object if the velocity is equal to the length of the segment?"
-- peculiar constraint

"An object moves along a surface of unknown concavity, what constraint does conservation of energy impose?"
-- abstract reasoning
-- no envisionment possible without total information

"Do objects collide?"
-- explanation

"The acceleration of the object on a straight section of slope x is y."
-- redundant information is ignored
-- possible contradictions are then also ignored

7.2 Extensions to NEWTON

In the previous section we have seen where many of NEWTON's limitations lie. The most apparent lack of knowledge occurs in the planning ability of the QAs. Unfortunately we do not yet know what this planning ability should be like. For that reason research should continue in attempting to broaden the knowledge in NEWTON so that more plans can be constructed and these plans investigated to see what kind of general planning ability is needed.

A number of the problems of 7.1 were unsolvable mainly because NEWTON could not accept artificial constraints. By creating a top level RALCM through which all of the usual top level RALCMs are invoked, external constraints can be at least quantitatively handled. A slight technical difficulty is present in identifying the modifiers of the variables but the DEFINE-PROBLEM forms can be developed to handle these more general constraints.

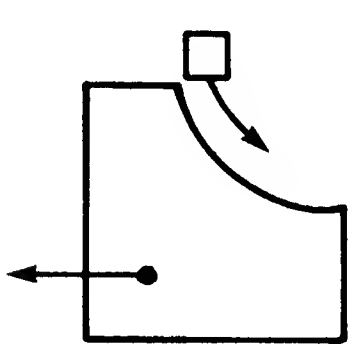
A deeper understanding of mechanics requires a knowledge of calculus. This is an obvious addition to make in so far as the major modifications involve the RALCMs and the mathematical expertise. The QAs remain exactly the same whether integration is possible or not. RALCMs already understand operators, differentiation and integration can be introduced without ever affecting the QAs. SOLVE when inverting operators would have to change integration into differentiation and differentiation into integration with the addition of suitable constant of integration (identifying the limits of integration).

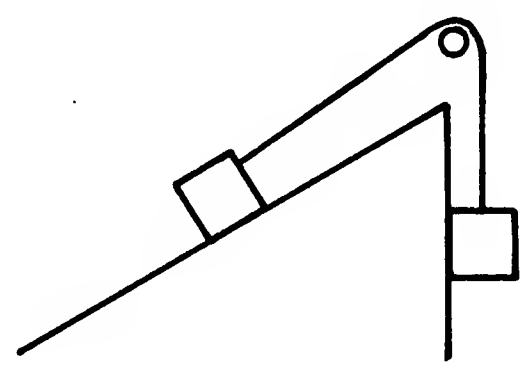
Another natural extension is that of comparison. Examples of this are comparing two paths to see which is faster, or modifying a problem and asking how that affects the final result. This requires both the introduction of partial differential equations and knowledge about examining the plans for a completed problem. Consider the example of the path over the hill discussed in section 4.5: would changing the height make any difference? The answer to that question is qualitative: the final velocity does not change but if the height is made high enough the object will fail to make it over the peak. With such extensions the example of 2.4 can be handled.

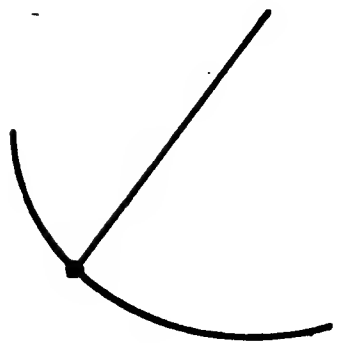
The structure of the envisioner is sufficiently powerful to make some simple additions. Trajectories and collisions with surfaces can easily be envisioned.

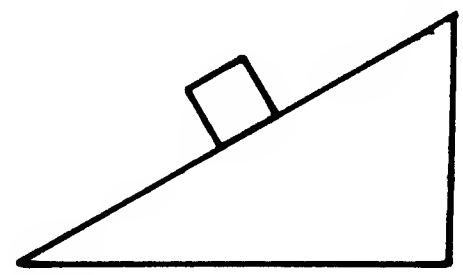
7.3 The Roller Coaster World in Mechanics

Physics problems that lie within the roller coaster world number no more than a dozen in any text. However, the roller coaster world is a subdomain of almost every mechanics problem. Any problem that involves any kind of sliding or constrained motion requires knowledge from the roller coaster world. Consider the following sequence of diagrams and the problems they suggest. None of the these problems can be understood without knowledge of the roller coaster world.









These problems require much more physics knowledge. An investigation into the knowledge required to understand these other domains and how knowledge of different domains interfaces with each other is required.

7.4 Principles for Representing Knowledge

NEWTON has tried to represent knowledge about physics. In the process of this research many issues about representing knowledge about formal domains has become evident. These issues are present in such diverse areas as network theory and thermodynamics.

QUANTITATIVE KNOWLEDGE:

Although knowledge that is explicitly given in a mechanics textbook must be represented, other, more intuitive, knowledge about mechanic is also required.

QUALITATIVE KNOWLEDGE:

This more intuitive knowledge not involved with equations must be represented if the quantitative knowledge can be ever be effectively used. One major part of this knowledge is the ability to qualitatively simulate or envision the event.

REDUNDANCY:

The necessity of representing the same information in many different forms results in the problem of knowing too much. Knowing too much does not so much lead to excessive computation as it does to undesired redundancy. For example if NEWTON knew about the rectangular and polar representation for a circle it would think the radius of any circle was derivable if no information about the circle was given at all. (NEWTON would set the problem in one coordinate scheme, change coordinates, substitute in for the polar form of the circle and find simultaneity for the radius, while the equations, if solved, would yield the redundant solution $r = r$.)

PLANNING:

One of the biggest problems which has already been discussed at length is the necessity for planning. Envisioning provides a great deal of planning, however, as we saw with the problem with heights, it is not enough. RALCM presents a number of ideas that would make planning easier. The envisioner and QAs make tests on the data base for conditions to determine which strategy to apply. Similarly RALCMs test conditions to see if their knowledge can be used by seeing if there are sufficient knowns to apply their equations. When a test fails within the QAs or the envisioner they just move on to another test while RALCM records the failing test as a complaint which when resolved causing computation to continue where the complaint left off. If the QAs and the envisioner could do this it would add greatly add to the flexibility of the plans that could be generated.

7.5 On the Relationship Between Mathematics and Physics

In many senses this research was an investigation into the relationship between mathematics and physics (or just mathematics applied to some domain). One of the main observations in chapter 1 was that classical science does indeed have a representation for encoding knowledge, however, it is only a quantitative representation and is incapable of encoding any qualitative knowledge. Although that was a starting point for this research, in retrospect this was a somewhat misleading observation.

NEWTON uses an equation as a constraint expression: given $f(x_1, \ldots, x_n)=0$ this relation is used to determine any x_i if all the other x_i's are known. There is much more information in an expression! If all we are interested in is solving a set of equations looking at constraint expressions may be a valid perspective. However, we are solving a physical problem in which a duality exists between the mathematical structure of the equations and the actual physical situation we have thrown away most of the information. To the the sophisticated student this duality is very clear and the mathematical equation is far more than a constraint expression. For him, the expression encodes a great deal of qualitative knowledge and every mathematical manipulation of the

expressions reflects some feature of the physical situation. These are matters for further research.

One of the reasons that NEWTON made such a sharp distinction between the mathematical symbol manipulation and the quantitative knowledge was the availablity of MACSYMA. MACSYMA provides a collection of operators which, unfortunately, must be regarded as black boxes. This made it impossible to study the duality between the mathematical manipulations and the physical situations. The final point is that the mathematics of a domain is so intimately connected with the qualitative understanding of the domain that both must be studied together and any attempt to separate them at a clean boundary will be doomed to failure

REFERENCES:

<Bobrow, 68>
Bobrow, G.D., "Natural Language Input for a Computer Problem Solving System", in Minsky (ed.), *Semantic Information Processing*, Cambridge: M.I.T. Press, 1968.

<Brown, 74>
Brown, A.L., "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures -- a Proposal for Research", Artificial Intelligence Laboratory, WP-61, Cambridge: M.I.T., 1974.

<Brown *et.al.*, 74>
Brown, John Seely, Richard R. Burton and Alan G. Bell, *SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An example of AI in CAI)*, Report 2790, A.I. Report 12, Cambridge: Bolt, Beranek and Newman, 1974.

<Charniak, 68>
Charniak, E., *CARPS, A Program which solves Calculus Word Problems*, Project MAC TR-51, Cambridge: M.I.T., 1971.

<Charniak, 71>
Charniak, E., "A Note on an Electrostatics Problem Solver", (unpublished), Artificial Intelligence Laboratory, Cambridge: M.I.T, 1971.

<Galileo, 60>
Galileo, G., *On Motion & On Mechanics*, 1600, Drabkin & Drake (translators), Madison: The University of Wisconsin Press, 1960.

<den Hartog, 48>
den Hartog, J.P., *Mechanics*, New York: McGraw-Hill, 1948.

<Hewitt, 71>
Hewitt, Carl, "Description and Theoretical Analysis (using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", Artificial Intelligence Laboratory, AIM-251, Cambridge: M.I.T., 1971.

<Kleppner & Kolenkow, 73>
Kleppner, Daniel, and Robert J. Kolenkow, *An Introduction to Mechanics*, New York: McGraw-Hill, 1973.

<Levinson, 61>
Levinson, I.J., *Introduction to Mechanics*, Englewood Cliffs: Prentice-Hall, 1961.

<Mach, 60>
Mach, E., *The Science of Mechanics*, Lasalle: Open Court, 1960.

<Mathlab, 74>
Mathlab Group, "MACSYMA Reference Manual", Cambridge: M.I.T., 1974.

<McDermott & Sussman, 74>
McDermott, Drew, and Gerald Jay Sussman, "The Conniver Reference Manual", Artificial Intelligence Laboratory, AIM-259a, Cambridge: M.I.T., 1974.

<Minsky M., 74>
Minsky, Marvin, "Frame-Systems: A Framework for Representation of Knowledge", Artificial Intelligence Laboratory, AIM-306, Cambridge: M.I.T, 1973.

<Moses, 67>
Moses, Joel; "Symbolic Integration", Project MAC, TR-47, Cambridge: M.I.T, 1967.

<Polya, 62>
Polya, George, "Mathematical Discovery", vol. I, John Wiley & Sons, Inc., 1962.

<Purcell, 65>
Purcell, M.E., *Electricity and Magnetism*, New York: McGraw-Hill, 1965.

<Sussman & Brown, 74>
Sussman, G.J., and A.L.Brown, "Localization of Failures in Radio Circuits a Study in Causal and Teleological Reasoning", Artificial Intelligence Laboratory, AIM-319, Cambridge: M.I.T., 1974.

<Sussman & Stallman, 75>
Sussman, G.J., and R.M.Stallman, "Heuristic Techniques in Computer Aided Circuit Analysis", Artificial Intelligence Laboratory, AIM-328, Cambridge: M.I.T., 1975.